A Simplified IoT Architecture

Although considerable differences exist between the aforementioned reference models, they each approach IoT from a layered perspective, allowing development of technology and standards somewhat independently at each level or domain. The commonality between these frameworks is that they all recognize the interconnection of the IoT endpoint devices to a network that transports the data where it is ultimately used by applications, whether at the data center, in the cloud, or at various management points throughout the stack.

It is not the intention of this book to promote or endorse any one specific IoT architectural framework. In fact, it can be noted that IoT architectures may differ somewhat depending on the industry use case or technology being deployed, and each has merit in solving the IoT heterogeneity problem discussed earlier. Thus, in this book we present an IoT framework that highlights the fundamental building blocks that are common to most IoT systems and which is intended to help you in designing an IoT network. This framework is presented as two parallel stacks: The IoT Data Management and Compute Stack and the Core IoT Functional Stack. Reducing the framework down to a pair of three-layer stacks in no way suggests that the model lacks the detail necessary to develop a sophisticated IoT strategy. Rather, the intention is to simplify the IoT architecture into its most basic building blocks and then to use it as a foundation to understand key design and deployment principles that are applied to industry-specific use cases. All the layers of more complex models are still covered, but they are grouped here in functional blocks that are easy to understand. Figure 2-6 illustrates the simplified IoT model presented in this book.

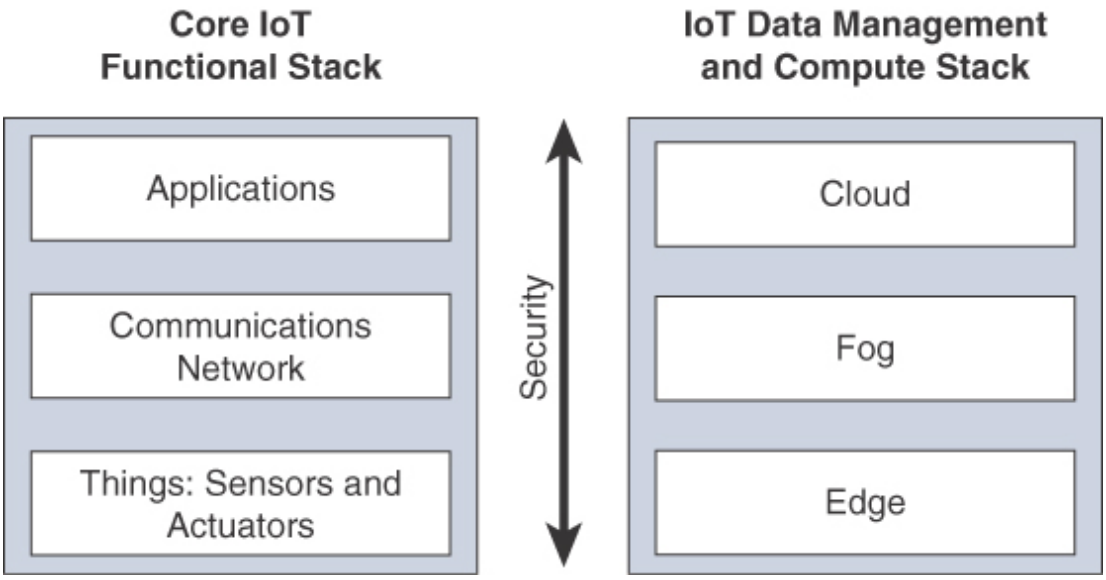

Figure 2-6 *Simplified IoT Architecture*

Nearly every published IoT model includes core layers similar to those shown on the left side of Figure 2-6, including "things," a communications network, and applications. However, unlike other models, the framework presented here separates the core IoT and data management into parallel and aligned stacks, allowing you to carefully examine the functions of both the network and the applications at each stage of a complex IoT system. This separation gives you better visibility into the functions of each layer.

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. Of course, such a simple architecture needs to be expanded on. The network communications layer of the IoT stack itself involves a significant amount of detail and incorporates a vast array of technologies. Consider for a moment the heterogeneity of IoT sensors and the many different ways that exist to connect them to a network. The network communications layer needs to consolidate these together, offer gateway and backhaul technologies, and ultimately bring the data back to a central location for analysis and processing.

Many of the last-mile technologies used in IoT are chosen to meet the specific requirements of the endpoints and are unlikely to ever be seen in the IT domain. However, the network between the gateway and the data center is composed mostly of traditional technologies that experienced IT professionals would quickly recognize. These include tunneling and VPN technologies, IP-based quality of service (QoS), conventional Layer 3 routing protocols such

as BGP and IP-PIM, and security capabilities such as encryption, access control lists (ACLs), and firewalls.

Unlike with most IT networks, the applications and analytics layer of IoT doesn't necessarily exist only in the data center or in the cloud. Due to the unique challenges and requirements of IoT, it is often necessary to deploy applications and data management throughout the architecture in a tiered approach, allowing data collection, analytics, and intelligent controls at multiple points in the IoT system. In the model presented in this book, 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). The IoT Data Management and Compute Stack is examined in greater detail later in this chapter. Figure 2-7 highlights an expanded view of the IoT architecture presented in this book.

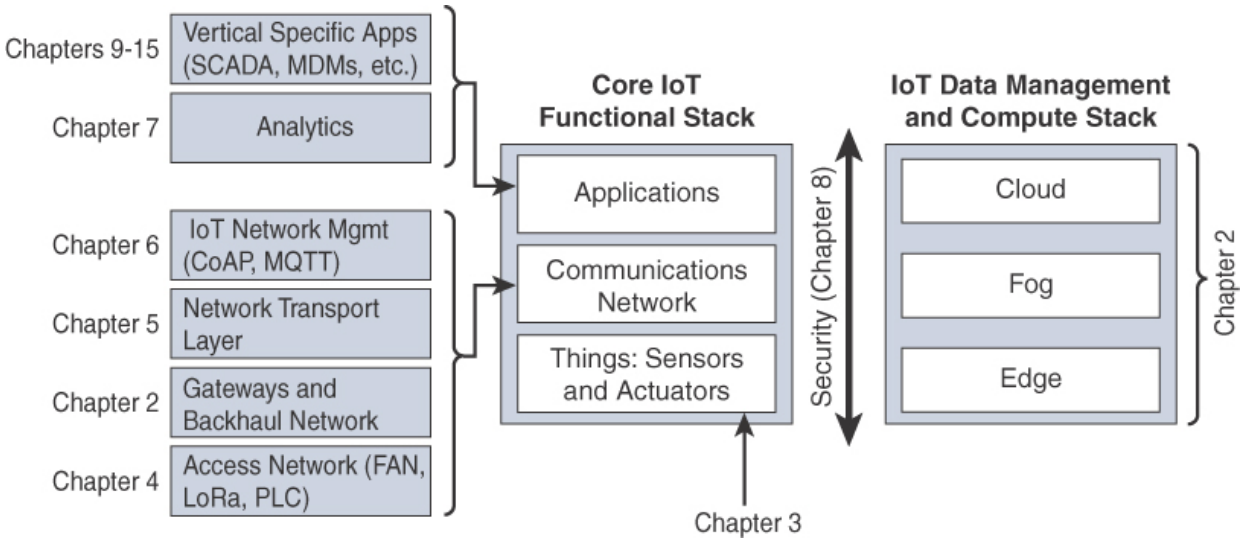

Figure 2-7 *Expanded View of the Simplified IoT Architecture*

As shown in Figure 2-7, 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.

You will notice that security is central to the entire architecture, both from network connectivity and data management perspectives. The chapters in Part II, "Engineering IoT Networks," discuss security at each layer. Chapter 8 is dedicated to the subject of securing IoT systems. The industry chapters in Part III, "IoT in Industry," highlight how lessons learned in Parts I, "Introduction to IoT," and II can be applied to specific industries. Each of the Part III chapters examines the issue of IoT security for a particular sector.

The architectural framework presented in Figure 2-7 reflects the flow of the chapters in this book. To help navigate your way through this book, chapter numbers are highlighted next to the various layers of the stack.

The remainder of this chapter provides a high-level examination of each layer of this model and lays the foundation for a detailed examination of the technologies involved at each layer 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. These actions can be self-contained (that is, the smart object does not rely on external systems for its actions); however, in most cases, the "thing" interacts with an external system to report information that the smart object collects, to exchange with other objects, or to interact with a management platform. In this case, the management platform can be used to process data collected from the smart object and also guide the behavior of the smart object. From an architectural standpoint, several components have to work together for an IoT network to be operational:

- **"Things" layer:** At this layer, the physical devices need to fit the constraints of the environment in which they are deployed while still being able to provide the information needed.

- **Communications network layer:** When smart objects are not self-contained, they need to communicate with an external system. In many cases, this communication uses a wireless technology. This layer has four sublayers:
  - **Access network sublayer:** The last mile of the IoT network is the access network. This is typically made up of wireless technologies such as 802.11ah, 802.15.4g, and LoRa. The sensors connected to the access network may also be wired.
  - **Gateways and backhaul network sublayer:** A common communication system organizes multiple smart objects in a given area around a common gateway. The gateway communicates directly with the smart objects. The role of the gateway is to forward the collected information through a longer-range medium (called the backhaul) to a headend central station where the information is processed. This information exchange is a Layer 7 (application) function, which is the reason this object is called a gateway. On IP networks, this gateway also forwards packets from one IP network to another, and it therefore acts as a router.
  - **Network transport sublayer:** For communication to be successful, network and transport layer protocols such as IP and UDP must be implemented to support the variety of devices to connect and media to use.
  - **IoT network management sublayer:** Additional protocols must be in place to allow the headend applications to exchange data with the sensors. Examples include CoAP and MQTT.
- **Application and analytics layer:** At the upper layer, an application needs to process the collected data, not only to control the smart objects when necessary, but to make intelligent decision based on the information collected and, in turn, instruct the "things" or other systems to adapt to the analyzed conditions and change their behaviors or parameters.

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

Layer 1: Things: Sensors and Actuators Layer

Most IoT networks start from the object, or "thing," that needs to be connected. Chapter 3, "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. There are myriad ways to classify smart objects. 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. Battery-powered things can be moved more easily than line-powered objects. However, batteries limit the lifetime and amount of energy that the object is allowed to consume, thus driving transmission range and frequency.
- **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 (for example, a viscosity sensor moved from batch to batch in a chemical plant) or because it is attached to a moving object (for example, a location sensor on moving goods in a warehouse or factory floor). The frequency of the movement may also vary, from occasional to permanent. The range of mobility (from a few inches to miles away) often drives the possible power source.
- **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. Higher frequencies drive higher energy consumption, which may create constraints on the possible power source (and therefore the object mobility) and the transmission range.
- **Simple or rich data:** This classification is based on the quantity of data exchanged at each report cycle. A humidity sensor in a field may report a simple daily index value (on a binary scale from 0 to 255), while an engine sensor may report hundreds of parameters, from temperature to pressure, gas velocity, compression speed, carbon index, and many others. Richer data typically drives higher power consumption. This classification is often combined with the previous to determine the

object data throughput (low throughput to high throughput). You may want to keep in mind that throughput is a combined metric. A medium-throughput object may send simple data at rather high frequency (in which case the flow structure looks continuous), or may send rich data at rather low frequency (in which case the flow structure looks bursty).

- **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. The assumption is that your phone needs to be at visual distance for you to consult the reported data on the phone screen. If the phone is far away, you typically do not use it, and reporting data from the band to the phone is not necessary. By contrast, a moisture sensor in the asphalt of a road may need to communicate with its reader several hundred meters or even kilometers away.
- **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. An oil pipeline may utilize a single sensor at key locations every few miles. By contrast, telescopes like the SETI Colossus telescope at the Whipple Observatory deploy hundreds, and sometimes thousands, of mirrors over a small area, each with multiple gyroscopes, gravity, and vibration sensors.

From a network architectural standpoint, your initial task is to determine which technology should be used to allow smart objects to communicate. This determination depends on the way the "things" are classified. However, some industries (such as manufacturing and utilities) may include objects in various categories, matching different needs. Figure 2-8 provides some examples of applications matching the combination of mobility and throughput requirements.

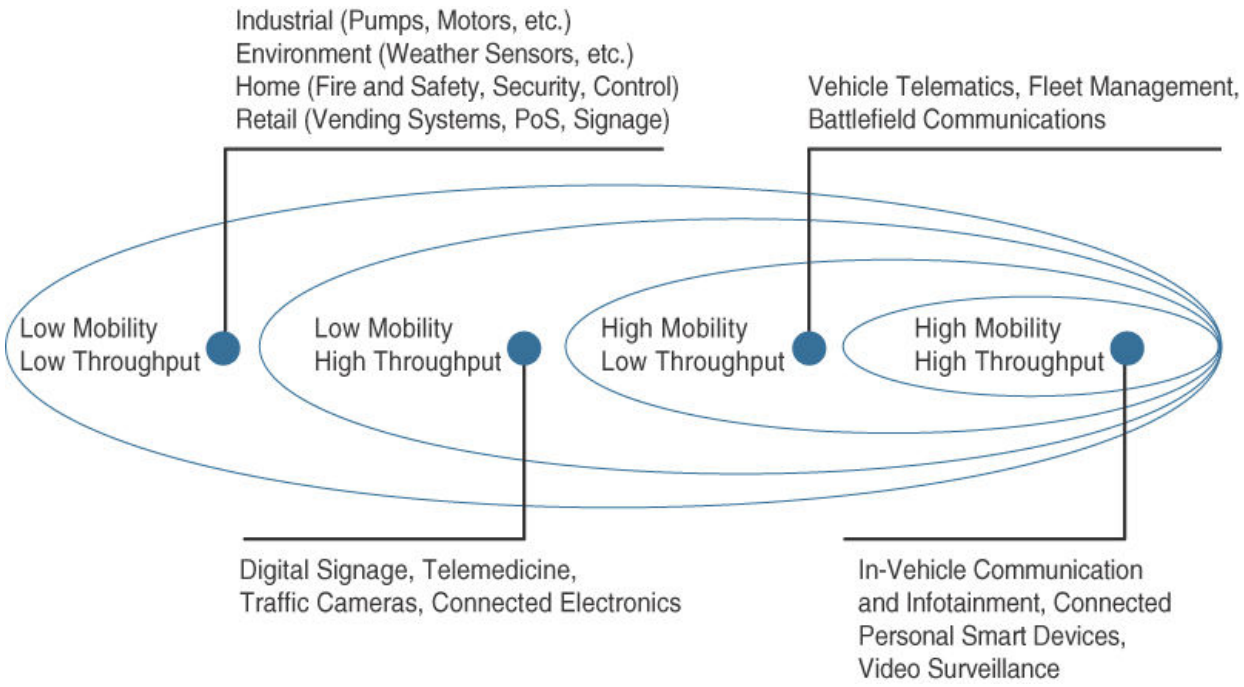

Figure 2-8 *Example of Sensor Applications Based on Mobility and Throughput*

The categories used to classify things can influence other parameters and can also influence one another. For example, a battery-operated highly mobile object (like a heart rate monitor, for example) likely has a small form factor. A small sensor is easier to move or integrate into its environment. At the same time, a small and highly mobile smart object is unlikely to require a large antenna and a powerful power source. This constraint will limit the transmission range and, therefore, the type of network protocol available for its connections. The criticality of data may also influence the form factor and, therefore, the architecture. For example, a missing monthly report from an asphalt moisture sensor may simply flag an indicator for sensor (or battery) replacement. A multi-mirror gyroscope report missing for more than 100 ms may render the entire system unstable or unusable. These sensors either need to have a constant source of power (resulting in limited mobility) or need to be easily accessible for battery replacement (resulting in limited transmission range). A first step in designing an IoT network is to examine the requirements in terms of mobility and data transmission (how much data, how often).

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.

Temperature variances are an easily understood metric. The cause for the variance is easily attributed to external weather forces and internal operating conditions. Remote external locations, such as those associated with mineral extraction or pipeline equipment can span from the heat of the Arabian Gulf to the cold of the Alaskan North Slope. Controls near the furnaces of a steel mill obviously require heat tolerance, and controls for cold food storage require the opposite. In some cases, these controls must handle extreme fluctuations as well. These extremes can be seen within a single deployment. For example, portions of the Tehachapi, California, wind farms are located in the Mojave Desert, while others are at an altitude of 1800 m in the surrounding mountains. As you can imagine, the wide variance in temperature takes a special piece of hardware that is capable of withstanding such harsh environments.

Humidity fluctuations can impact the long-term success of a system as well. Well heads residing in the delta of the Niger River will see very different conditions from those in the middle of the Arabian Desert. In some conditions, the systems could be exposed to direct liquid contact such as may be found with outdoor wireless devices or marine condition deployments.

Less obvious are the operating extremes related to kinetic forces. Shock and vibration needs vary based on the deployment scenario. In some cases, the focus is on low-amplitude but constant vibrations, as may be expected on a bushing-mounted manufacturing system. In other cases, it could be a sudden acceleration or deceleration, such as may be experienced in peak ground acceleration of an earthquake or an impact on a mobile system such as high-speed rail or heavy-duty earth moving equipment.

Solid particulates can also impact the gear. Most IT environments must contend with dust build-up that can become highly concentrated due to the effect of cooling fans. In less-controlled IT environments, that phenomenon can be accelerated due to higher concentrations of particulates. A deterrent to particulate build-up is to use fanless cooling, which necessitates a higher surface area, as is the case with heat transfer fins.

Hazardous location design may also cause corrosive impact to the equipment. Caustic materials can impact connections over which power or communications travel. Furthermore, they can result in reduced thermal efficiency by potentially coating the heat transfer surfaces.

In some scenarios, the concern is not how the environment can impact the equipment but how the equipment can impact the environment. For example, in a scenario in which volatile gases may be present, spark suppression is a critical design criterion.

There is another class of device differentiators related to the external connectivity of the device for mounting or industrial function. Device mounting is one obvious difference between OT and IT environments. While there are rack mount environments in some industrial spaces, they are more frequently found among IT type assets. Within industrial environments, many compute and communication assets are placed within an enclosed space, such as a control cabinet where they will be vertically mounted on a DIN (Deutsches Institut für Normung) rail inside. In other scenarios, the devices might be mounted horizontally directly on a wall or on a fence.

In contrast to most IT-based systems, industrial compute systems often transmit their state or receive inputs from external devices through an alarm channel. These may drive an indicator light (stack lights) to display the status of a process element from afar. This same element can also receive inputs to initiate actions within the system itself.

Power supplies in OT systems are also frequently different from those commonly seen on standard IT equipment. A wider range of power variations are common attributes of industrial compute components. DC power sources are also common in many environments. Given the criticality of many systems, it is often required that redundant power supplies be built into the device itself. Extraneous power supplies, especially those not inherently mounted, are frowned upon, given the potential for accidental unplugging. In some

utility cases, the system must be able to handle brief power outages and still continue to operate.

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.

As IoT continues to grow exponentially, you will encounter a wide variety of applications and special use cases. For each of them, an access technology will be required. IoT sometimes reuses existing access technologies whose characteristics match more or less closely the IoT use case requirements. Whereas some access technologies were developed specifically for IoT use cases, others were not.

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.

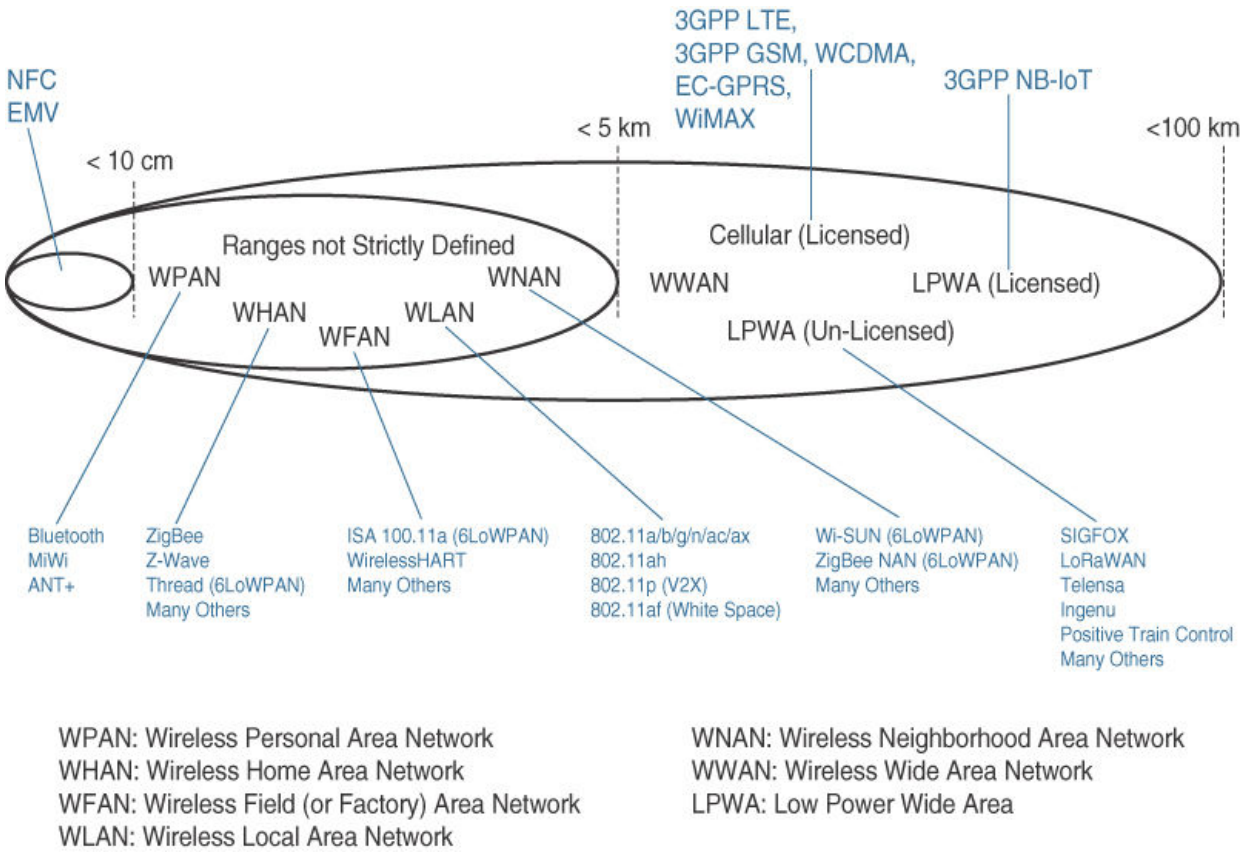

Figure 2-9 *Access Technologies and Distances*

Note that the ranges in Figure 2-9 are inclusive. For example, cellular is indicated for transmissions beyond 5 km, but you could achieve a successful cellular transmission at shorter range (for example, 100 m). By contrast, ZigBee is expected to be efficient over a range of a few tens of meters, but you would not expect a successful ZigBee transmission over a range of 10 km.

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). A FAN is sometimes viewed as a group of NANs, but some verticals see the FAN as a group of HANs or a group of smaller outdoor cells. As you can see, FAN and NAN may sometimes be used interchangeably. In most cases, the vertical context is clear enough to determine the grouping hierarchy.
- **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. Other networking classifications, such as MAN (metropolitan area network, with a range of up to a few kilometers) and WAN (wide area network, with a range of more than a few kilometers), are also commonly used.

Note that for all these places in the IoT network, a "W" can be added to specifically indicate wireless technologies used in that space. For example, HomePlug is a wired technology found in a HAN environment, but a HAN is often referred to as a WHAN (wireless home area network) when a wireless technology, like ZigBee, is used in that space.

Similar achievable distances do not mean similar protocols and similar characteristics. Each protocol uses a specific frame format and transmission technique over a specific frequency (or band). These characteristics introduce additional differences. For example, Figure 2-10 demonstrates four technologies representing WHAN to WLAN ranges and compares the throughput and range that can be achieved in each case. Figure 2-10 supposes that the sensor uses the same frame size, transmit power, and antenna gain. The slope of throughput degradation as distance increases varies vastly from one technology to the other. This difference limits the amount of data throughput that each technology can achieve as the distance from the sensor to the receiver increases.

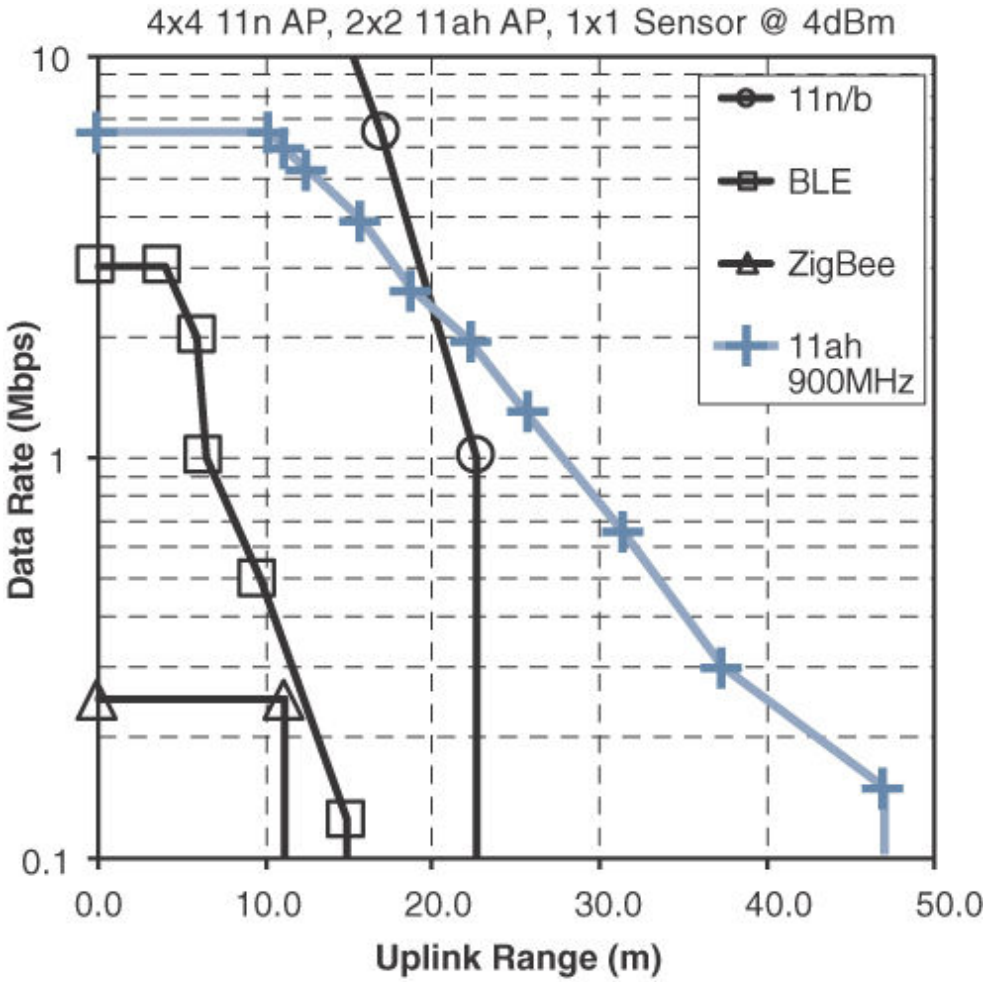

Figure 2-10 *Range Versus Throughput for Four WHAN to WLAN Technologies*

Increasing the throughput and achievable distance typically comes with an increase in power consumption. Therefore, after determining the smart object requirements (in terms of mobility and data transfer), a second step is to determine the target quantity of objects in a single collection cell, based on the transmission range and throughput required. This parameter in turn determines the size of the cell.

It may be tempting to simply choose the technology with the longest range and highest throughput. However, the cost of the technology is a third determining

factor. Figure 2-11 combines cost, range, power consumption, and typical available bandwidth for common IoT access technologies.

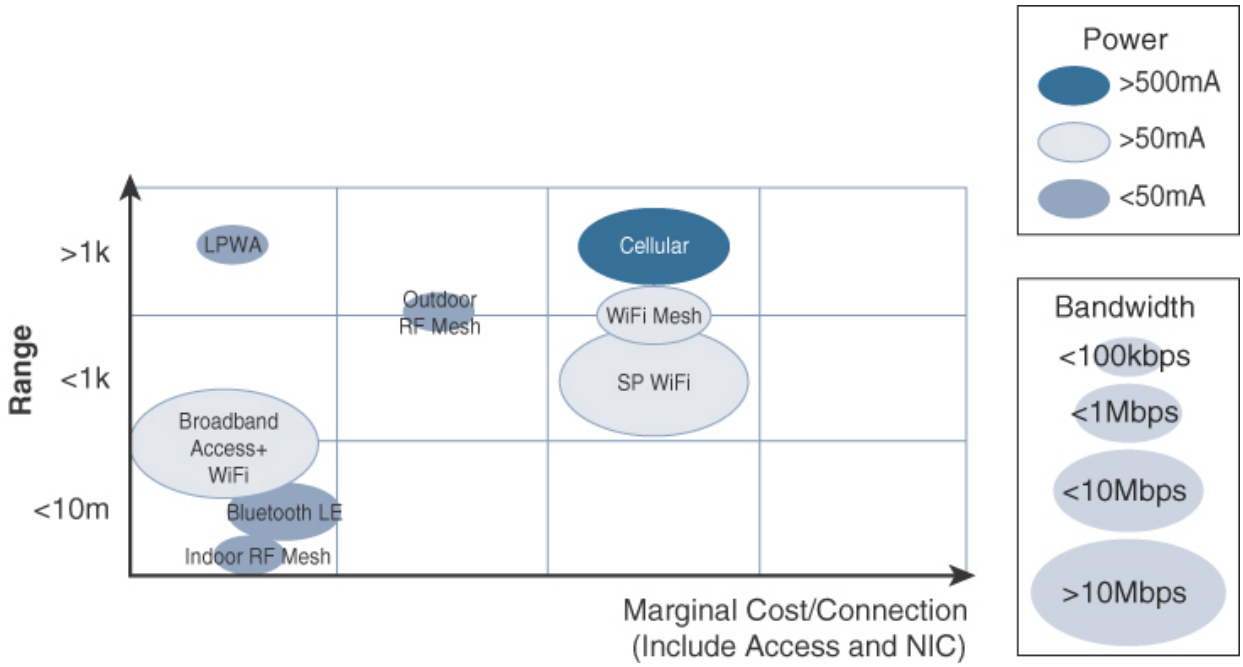

Figure 2-11 *Comparison Between Common Last-Mile Technologies in Terms of Range Versus Cost, Power, and Bandwidth*

The amount of data to carry over a given time period along with correlated power consumption (driving possible limitations in mobility and range) determines the wireless cell size and structure.

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

- **Point-to-point topologies:** These topologies allow one point to communicate with another point. This topology in its strictest sense is uncommon for IoT access, as it would imply that a single object can communicate only with a single gateway. However, several technologies are referred to as "point-to-point" when each object establishes an individual session with the gateway. The "point-to-point" concept, in that case, often refers to the communication structure more than the physical topology.
- **Point-to-multipoint topologies:** These topologies allow one point to communicate with more than one other point. Most IoT technologies where one or more than one gateways communicate with multiple smart objects are in this category. However, depending on the features

available on each communicating mode, several subtypes need to be considered. A particularity of IoT networks is that some nodes (for example, sensors) support both data collection and forwarding functions, while some other nodes (for example, some gateways) collect the smart object data, sometimes instruct the sensor to perform specific operations, and also interface with other networks or possibly other gateways. For this reason, some technologies categorize the nodes based on the functions (described by a protocol) they implement.

An example of a technology that categorizes nodes based on their function is IEEE 802.15.4, which is covered in depth in Chapter 4. Although 802.15.4 is used as an example in this section, the same principles may apply to many other technologies. Applications leveraging IEEE 802.15.4 commonly rely on the concept of an end device (a sensor) collecting data and transmitting the data to a collector. Sensors need to be small and are often mobile (or movable). When mobile, these sensors are therefore commonly battery operated.

To form a network, a device needs to connect with another device. When both devices fully implement the protocol stack functions, they can form a peer-to-peer network. However, in many cases, one of the devices collects data from the others. For example, in a house, temperature sensors may be deployed in each room or each zone of the house, and they may communicate with a central point where temperature is displayed and controlled. A room sensor does not need to communicate with another room sensor. In that case, the control point is at the center of the network. The network forms a star topology, with the control point at the hub and the sensors at the spokes.

In such a configuration, the central point can be in charge of the overall network coordination, taking care of the beacon transmissions and connection to each sensor. In the IEEE 802.15.4 standard, the central point is called a *coordinator* for the network. With this type of deployment, each sensor is not intended to do anything other than communicate with the coordinator in a master/slave type of relationship. The sensor can implement a subset of protocol functions to perform just a specialized part (communication with the coordinator). Such a device is called a reduced-function device (RFD). An RFD cannot be a coordinator. An RFD also cannot implement direct communications to another RFD.

The coordinator that implements the full network functions is called, by contrast, a full-function device (FFD). An FFD can communicate directly with

another FFD or with more than one FFD, forming multiple peer-to-peer connections. Topologies where each FFD has a unique path to another FFD are called cluster tree topologies. FFDs in the cluster tree may have RFDs, resulting in a cluster star topology. Figure 2-12 illustrates these topologies.

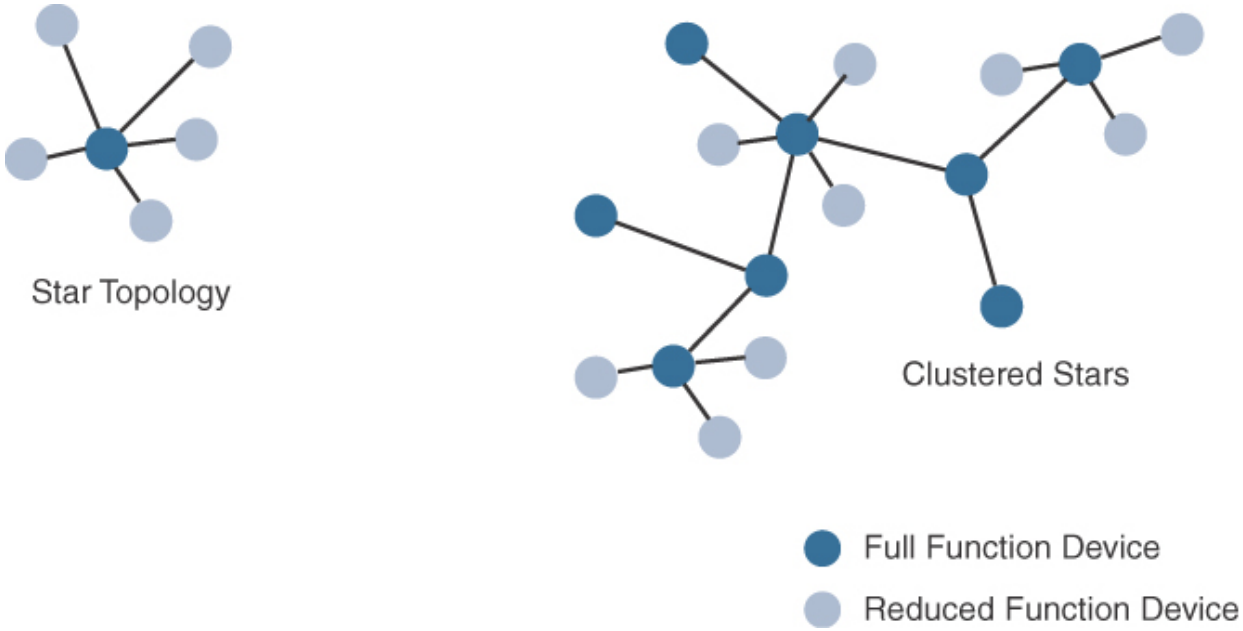

Figure 2-12 *Star and Clustered Star Topologies*

Other point-to-multipoint technologies allow a node to have more than one path to another node, forming a mesh topology. This redundancy means that each node can communicate with more than just one other node. This communication can be used to directly exchange information between nodes (the receiver directly consumes the information received) or to extend the range of the communication link. In this case, an intermediate node acts as a relay between two other nodes. These two other nodes would not be able to communicate successfully directly while respecting the constraints of power and modulation dictated by the PHY layer protocol. Range extension typically comes at the price of slower communications (as intermediate nodes need to spend time relaying other nodes' messages). An example of a technology that implements a mesh topology is Wi-Fi mesh.

Another property of mesh networks is redundancy. The disappearance of one node does not necessarily interrupt network communications. Data may still be relayed through other nodes to reach the intended destination.

Figure 2-13 shows a mesh topology. Nodes A and D are too far apart to communicate directly. In this case, communication can be relayed through

nodes B or C. Node B may be used as the primary relay. However, the loss of node B does not prevent the communication between nodes A and D. Here, communication is rerouted through another node, node C.

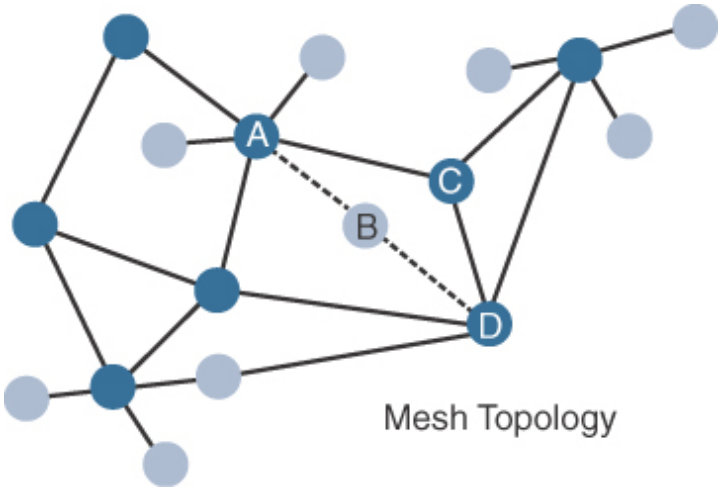

Figure 2-13 *Mesh Topology*

Note

Figure 2-13 shows a partial mesh topology, where a node can communicate with more than one other node, but not all nodes communicate directly with all other nodes. In a full mesh topology each node communicates with each other node. In the topology shown in Figure 2-13, which has 17 nodes, a full mesh structure would mean that each node would have 16 connections (one to each other node). Full mesh structures are computationally expensive (as each node needs to maintain a connection to each other node). In the IoT space, full mesh deployments are uncommon. In most cases, information has to travel to a target destination rather than being directly distributed to all other nodes. Full mesh topologies also limit the acceptable distance between nodes (as all nodes must be in range of all other nodes).

Note

Do not confuse *topology* and *range*. Topology describes the organization of the nodes, while range is dictated by factors such as the frequency or operation, the signal structure, and operational bandwidth. For example, both IEEE 802.15.4 and LoRaWAN implement star topologies, but the range of IEEE 802.15.4 is a few tens of meters, while LoRaWAN can achieve a successful signal over many kilometers. The bandwidth and signal structure (modulation) are very different. Figure 2-11 helps you compare the use cases and implementation considerations (range, cost, available bandwidth) for common IoT access technologies. Chapter 4 describes in detail IEEE 802.15.4, LTE, LoRaWAN, and other competing IoT technologies. However, keep in mind that many technologies that were not initially designed for IoT usage can be leveraged by specific applications. For example, remote sites may need to leverage satellite communications when standard IoT wireless technologies cannot achieve the range required. Also, the adoption of technology can vary widely over time, based on use cases, technology maturity, and other factors. For example, cellular technologies were initially designed for voice communications. The burst of data traffic that accompanied the explosion of mobile devices in the early 2000s brought the development of enhanced standards for cellular communications (with LTE). In turn, this enhancement allowed LTE to grow rapidly as a major connection technology for FANs.

Gateways and Backhaul Sublayer

Data collected from a smart object may need to be forwarded to a central station where data is processed. As this station is often in a different location from the smart object, data directly received from the sensor through an access technology needs to be forwarded to another medium (the backhaul) and transported to the central station. The gateway is in charge of this inter-medium communication.

In most cases, the smart objects are static or mobile within a limited area. The gateway is often static. However, some IoT technologies do not apply this

model. For example, dedicated short-range communication (DSRC) allows vehicle-to-vehicle and vehicle-to-infrastructure communication. In this model, the smart object's position relative to the gateway is static. The car includes sensors and one gateway. Communication between the sensors and the gateway may involve wired or wireless technologies. Sensors may also be integrated into the road infrastructure and connect over a wired or wireless technology to a gateway on the side of the road. A wireless technology (DSRC operates in the upper 5 GHz range) is used for backhaul communication, peer-to-peer, or mesh communication between vehicles.

In the DSRC case, the entire "sensor field" is moving along with the gateway, but the general principles of IoT networking remain the same. The range at which DSRC can communicate is limited. Similarly, for all other IoT architectures, the choice of a backhaul technology depends on the communication distance and also on the amount of data that needs to be forwarded. When the smart object's operation is controlled from a local site, and when the environment is stable (for example, factory or oil and gas field), Ethernet can be used as a backhaul. In unstable or changing environments (for example, open mines) where cables cannot safely be run, a wireless technology is used. Wi-Fi is common in this case, often with multiple hops between the sensor field and the operation center. Mesh is a common topology to allow communication flexibility in this type of dynamic environment.

However, throughput decreases as node-to-node distance increases, and it also decreases as the number of hops increases. In a typical Wi-Fi mesh network, throughput halves for each additional hop. Some technologies, like 802.11ah, implement Wi-Fi in a lower band (lower than 1 GHz instead of 2.4 GHz/5 GHz for classical Wi-Fi) with special provisions adapted to IoT, to achieve a longer range (up to about 2 km). Beyond that range, other technologies are needed.

WiMAX (802.16) is an example of a longer-range technology. WiMAX can achieve ranges of up to 50 kilometers with rates of up to 70 Mbps. Obviously, you cannot achieve maximum rate at maximum range; you could expect up to 70 Mbps at short range and 2 to 3 Mbps at maximum range. 802.16d (also called Fixed WiMAX) describes the backhaul implementation of the protocol. Improvements to this aspect have been published (802.16.1), but most WiMAX networks still implement a variation of 802.16d. 802.16 can operate in unlicensed bands, but its backhaul function is often deployed in more-

reliable licensed bands, where interferences from other systems are better controlled.

As licensed bands imply the payment of a usability fee, other cellular technologies also grew as competitive solutions for the backhaul part to achieve similar range. The choice of WiMAX or a cellular technology depends on the vertical and the location (local preferences, local costs). Chapter 4 offers an in-depth look at the most commonly deployed protocols for this segment, and Table 2-4 compares 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)

Table 2-4 *Architectural Considerations for WiMAX and Cellular Technologies*

Network Transport Sublayer

The previous section describes a hierarchical communication architecture in which a series of smart objects report to a gateway that conveys the reported data over another medium and up to a central station. However, practical implementations are often flexible, with multiple transversal communication paths. For example, consider the case of IoT for the energy grid. Your house may have a meter that reports the energy consumption to a gateway over a wireless technology. Other houses in your neighborhood (NAN) make the same report, likely to one or several gateways. The data to be transported is small and the interval is large (for example, four times per hour), resulting in a low-mobility, low-throughput type of data structure, with transmission distances up to a mile. Several technologies (such as 802.11ah, 802.15.4, or LPWA) can be used for this collection segment. Other neighborhoods may also connect the same way, thus forming a FAN.

For example, the power utility's headend application server may be regional, and the gateway may relay to a wired or wireless backhaul technology. The structure appears to be hierarchical. Practically, however, this IoT system may achieve more than basic upstream reporting. If your power consumption becomes unusually high, the utility headend application server may need on-demand reporting from your meter at short intervals to follow the consumption trend. From a standard vertical push model, the transport structure changes and becomes bidirectional (downstream pull model instead of upstream push).

Distribution automation (DA) also allows your meter to communicate with neighboring meters or other devices in the electrical distribution grid. With such communication, consumption load balancing may be optimized. For example, your air conditioning pulses fresh air at regular intervals. With DA, your neighbor's AC starts pulsing when your system pauses; in this way, the air in both houses is kept fresh, but the energy consumed from the network is stable instead of spiking up and down with uncoordinated start and stop points. Here again, the transport model changes. From a vertical structure, you are now changing to a possible mesh structure with multiple peer-to-peer exchanges.

Similarly, your smart meter may communicate with your house appliances to evaluate their type and energy demand. With this scheme, your washing machine can be turned on in times of lower consumption from other systems,

such as at night, while power to your home theater system will never be deprived, always turning on when you need it. Once the system learns your consumption pattern, charging of your electric car can start and stop at intervals to achieve the same overnight charge without creating spikes in energy demand. When these functions appear, the transport model changes again. A mesh system may appear at the scale of the house. More commonly, a partial mesh appears, with some central nodes connecting to multiple other nodes. Data may flow locally, or it may have to be orchestrated by a central application to coordinate the power budget between houses.

In this smart system, your car's charging system is connected to your energy account. As you plug into a public charging station, your car logs into the system to be identified and uniquely links to your account. At regular intervals, the central system may need to query all the charging stations for status update. The transport structure loses its vertical organization a bit more in this model, as you may be connecting from anywhere. In a managed environment, the headend system needs to upgrade the software on your meter, just as appliance vendors may need to update your oven or washing machine smart energy software. From a bottom-up data transport flow, you now implement top-down data flows.

This communication structure thus may involve peer-to-peer (for example, meter to meter), point-to-point (meter to headend station), point-to-multipoint (gateway or head-end to multiple meters), unicast and multicast communications (software update to one or multiple systems). In a multitenant environment (for example, electricity and gas consumption management), different systems may use the same communication pathways. This communication occurs over multiple media (for example, power lines inside your house or a short-range wireless system like indoor Wi-Fi and/or ZigBee), a longer-range wireless system to the gateway, and yet another wireless or wired medium for backhaul transmission.

To allow for such communication structure, a network protocol with specific characteristics needs to be implemented. The protocol needs to be open and standard-based to accommodate multiple industries and multiple media. Scalability (to accommodate thousands or millions of sensors in a single network) and security are also common requirements. IP is a protocol that matches all these requirements. The advantages of IP are covered in depth in Chapter 5.

The flexibility of IP allows this protocol to be embedded in objects of very different natures, exchanging information over very different media, including low-power, lossy, and low-bandwidth networks. For example, RFC 2464 describes how an IPv6 packet gets encapsulated over an Ethernet frame and is also used for IEEE 802.11 Wi-Fi. Similarly, the IETF 6LoWPAN working group specifies how IPv6 packets are carried efficiently over lossy networks, forming an "adaption layer" for IPv6, primarily for IoT networks. Chapter 4 provides more details on 6LoWPAN and its capabilities.

Finally, the transport layer protocols built above IP (UDP and TCP) can easily be leveraged to decide whether the network should control the data packet delivery (with TCP) or whether the control task should be left to the application (UDP). UDP is a much lighter and faster protocol than TCP. However, it does not guarantee packet delivery. Both TCP and UDP can be secured with TLS/SSL (TCP) or DTLS (UDP). Chapter 6 takes a closer look at TCP and UDP for IoT networks.

IoT Network Management Sublayer

IP, TCP, and UDP bring connectivity to IoT networks. Upper-layer protocols need to take care of data transmission between the smart objects and other systems. Multiple protocols have been leveraged or created to solve IoT data communication problems. Some networks rely on a push model (that is, a sensor reports at a regular interval or based on a local trigger), whereas others rely on a pull model (that is, an application queries the sensor over the network), and multiple hybrid approaches are also possible.

Following the IP logic, some IoT implementers have suggested HTTP for the data transfer phase. After all, HTTP has a client and server component. The sensor could use the client part to establish a connection to the IoT central application (the server), and then data can be exchanged. You can find HTTP in some IoT applications, but HTTP is something of a fat protocol and was not designed to operate in constrained environments with low memory, low power, low bandwidth, and a high rate of packet failure. Despite these limitations, other web-derived protocols have been suggested for the IoT space. One example is WebSocket. WebSocket is part of the HTML5 specification, and provides a simple bidirectional connection over a single connection. Some IoT solutions use WebSocket to manage the connection between the smart object and an external application. WebSocket is often

combined with other protocols, such as MQTT (described shortly) to handle the IoT-specific part of the communication.

With the same logic of reusing well-known methods, Extensible Messaging and Presence Protocol (XMPP) was created. XMPP is based on instant messaging and presence. It allows the exchange of data between two or more systems and supports presence and contact list maintenance. It can also handle publish/subscribe, making it a good choice for distribution of information to multiple devices. A limitation of XMPP is its reliance on TCP, which may force subscribers to maintain open sessions to other systems and may be a limitation for memory-constrained objects.

To respond to the limits of web-based protocols, another protocol was created by the IETF Constrained Restful Environments (CoRE) working group: Constrained Application Protocol (CoAP). CoAP uses some methods similar to those of HTTP (such as Get, Post, Put, and Delete) but implements a shorter list, thus limiting the size of the header. CoAP also runs on UDP (whereas HTTP typically uses TCP). CoAP also adds a feature that is lacking in HTTP and very useful for IoT: observation. Observation allows the streaming of state changes as they occur, without requiring the receiver to query for these changes.

Another common IoT protocol utilized in these middle to upper layers is Message Queue Telemetry Transport (MQTT). MQTT uses a broker-based architecture. The sensor can be set to be an MQTT publisher (publishes a piece of information), the application that needs to receive the information can be set as the MQTT subscriber, and any intermediary system can be set as a broker to relay the information between the publisher and the subscriber(s). MQTT runs over TCP. A consequence of the reliance on TCP is that an MQTT client typically holds a connection open to the broker at all times. This may be a limiting factor in environments where loss is high or where computing resources are limited.

Chapter 6 examines in more detail the various IoT application protocols, including CoAP and MQTT. From an architectural standpoint, you need to determine the requirements of your application protocol. Relying on TCP implies maintaining sessions between endpoints. The advantage of reliability comes with the cost of memory and processing resources consumed for session awareness. Relying on UDP delegates the control to the upper layers. You also need to determine the requirements for QoS with different priority levels between the various messages. Finally, you need to evaluate the

security of the IoT application protocol to balance the level of security provided against the overhead required. Chapter 8 describes how to evaluate the security aspect of IoT networks.

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. Chapter 7, "Data and Analytics for IoT," provides an in-depth analysis of the various application families. 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.

An example of control system architecture is SCADA. SCADA was developed as a universal method to access remote systems and send instructions. One example where SCADA is widely used is in the control and monitoring of remote terminal units (RTUs) on the electrical distribution grid.

Many advanced IoT applications include both analytics and control modules. In most cases, data is collected from the smart objects and processed in the analytics module. The result of this processing may be used to modify the behavior of smart objects or systems related to the smart objects. The control module is used to convey the instructions for behavioral changes. When evaluating an IoT data and analytics application, you need to determine the relative depth of the control part needed for your use case and match it against the type of analytics provided.

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. In this case, data processing can be very complex and may combine multiple changing values over complex algorithms. Data analytics can also monitor the IoT system itself. For example, a machine or robot in a factory can report data about its own movements. This data can be used by an analytics application to report degradation in the movement speeds, which may be indicative of a need to service the robot before a part breaks.
- **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. A similar consequence is that the control module cannot modify local object behaviors anymore.

Most analytics applications employ both data and network analytics modules. When architecting an IoT system, you need to evaluate the need for each one. Network analytics is necessary for connected systems. However, the depth of analysis depends on your use cases. A basic connectivity view may be enough if the smart objects report occasional status, without expectation for immediate action based on this report. Detailed analysis and trending about network performance are needed if the central application is expected to pilot in near-real-time connected systems.

Data analytics is a wider space with a larger gray area (in terms of needs) than network analytics. Basic systems analytics can provide views of the system state and state trend analysis. More advanced systems can refine the type of data collected and display additional information about the system. The type of collected data and processing varies widely with the use case.

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.

From an architectural standpoint, you can define static IoT networks where a clear list of elements to monitor and analytics to perform are determined. Such static systems are common in industrial environments where the IoT charter is about providing a clear view of the state of the operation. However, a smarter architectural choice may be to allow for an open system where the network is engineered to be flexible enough that other sensors may be added in the future, and where both upstream and downstream operations are allowed. This flexibility allows for additional processing of the existing sensors and also deeper and more efficient interaction with the connected objects. This enhanced data processing can result in new added value for businesses that are not envisioned at the time when the system is initially deployed.

An example of a flexible analytics and control application is Cisco Jasper, which provides a turnkey cloud-based platform for IoT management and monetization. Consider the case of vending machines deployed throughout a city. At a basic level, these machines can be connected, and sensors can be deployed to report when a machine is in an error state. A repair person can be sent to address the issue when such a state is identified. This type of alert is a time saver and avoids the need for the repair team to tour all the machines in turn when only one may be malfunctioning.

This alert system may also avoid delay between the time when a machine goes into the error state and the time when a repair team visits the machine location. With a static platform, this use case is limited to this type of alert. With a flexible platform like Cisco Jasper, new applications may be imagined and developed over time. For example, the machine sensors can be improved to also report when an item is sold. The central application can then be enhanced to process this information and analyze what item is most sold, in what location, at what times. This new view of the machines may allow for an optimization of the items to sell in machines in a given area. Systems may be implemented to adapt the goods to time, season, or location—or many other parameters that may have been analyzed. In short, architecting open systems opens the possibility for new applications.

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.

Fundamentally, smart services use IoT and aim for efficiency. For example, sensors can be installed on equipment to ensure ongoing conformance with regulations or safety requirements. This angle of efficiency can take multiple forms, from presence sensors in hazardous areas to weight threshold violation detectors on trucks.

Smart services can also be used to measure the efficiency of machines by detecting machine output, speed, or other forms of usage evaluation. Entire operations can be optimized with IoT. In hospitality, for example, presence and motion sensors can evaluate the number of guests in a lobby and redirect personnel accordingly. The same type of action can be taken in a store where a customer is detected as staying longer than the typical amount of time in

front of a shelf. Personnel can be deployed to provide assistance. Movement of people and objects on factory floors can be analyzed to optimize the production flow.

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. An even smarter system can communicate with other systems in the house, learn the human movement pattern, and anticipate the presence of a human, turning on the light just before the person enters the room. An even smarter system can use smarter sensors that analyze multiple parameters to detect human mood and modify accordingly the light color to adapt to the learned preferences, or to convey either a more relaxing or a more dynamic environment.

Light bulbs are a simple example. By connecting to other systems in the house, efficiencies can be coordinated. For example, the house entry alarm system or the heating system can coordinate with the presence detector in a light bulb to adapt to detected changes. The alarm system can disable volumetric movement alarms in zones where a known person is detected. The heating system can adapt the temperature to human presence or detected personal preferences.

Similar efficiency can be extended to larger systems than a house. For example, smart grid applications can coordinate the energy consumption between houses to regulate the energy demand from the grid. We already mentioned that your washing machine may be turned on at night when the energy demand for heating and cooling is lower. Just as your air conditioning pulses can be coordinated with your neighbor's, your washing machine cycles can be coordinated with the appliances in your house and in the neighborhood to smooth the energy demand spikes on the grid.

Efficiency also applies to M2M communications. In mining environments, vehicles can communicate to regulate the flows between drills, draglines, bulldozers, and dump trucks, for example, making sure that a dump truck is always available when a bulldozer needs it. In smart cities, vehicles communicate. A traffic jam is detected and anticipated automatically by public transportation, and the system can temporarily reroute buses or regulate the number of buses servicing a specific line based on traffic and customer quantity, instantaneous or learned over trending.

Part III of this book provides detailed examples of how IoT is shaping specific industries. The lessons learned are always that architecting open IoT systems allows for increased efficiency over time. New applications and possibilities for an IoT system will appear in the upcoming years. When building an IoT network, you should make sure to keep the system open for the possibility of new smart objects and more traffic on the system.

IoT Data Management and Compute Stack

One of the key messages in the first two chapters of this book is that the massive scale of IoT networks is fundamentally driving new architectures. For instance, Figure 1-2 in Chapter 1 illustrates how the "things" connected to the Internet are continuing to grow exponentially, with a prediction by Cisco that by 2020 there will be more than 50 billion devices connected to some form of an IP network. Clearly, traditional IT networks are not prepared for this magnitude of network devices. However, beyond the network architecture itself, consider the data that is generated by these devices. If the number of devices is beyond conventional numbers, surely the data generated by these devices must also be of serious concern.

In fact, the data generated by IoT sensors is one of the single biggest challenges in building an IoT system. In the case of modern IT networks, the data sourced by a computer or server is typically generated by the client/server communications model, and it serves the needs of the application. In sensor networks, the vast majority of data generated is unstructured and of very little use on its own. For example, the majority of data generated by a smart meter is nothing more than polling data; the communications system simply determines whether a network connection to the meter is still active. This data on its own is of very little value. The real value of a smart meter is the metering data read by the meter management system (MMS). However, if you look at the raw polling data from a different perspective, the information can be very useful. For example, a utility may have millions of meters covering its entire service area. If whole sections of the smart grid start to show an interruption of connectivity to the meters, this data can be analyzed and combined with other sources of data, such as weather reports and electrical demand in the grid, to provide a complete picture of what is happening. This information can help determine whether the loss of connection to the meters is truly a loss of power or whether some other problem has developed in the grid. Moreover, analytics of this data can help

the utility quickly determine the extent of the service outage and repair the disruption in a timely fashion.

In most cases, the processing location is outside the smart object. A natural location for this processing activity is the cloud. Smart objects need to connect to the cloud, and data processing is centralized. One advantage of this model is simplicity. Objects just need to connect to a central cloud application. That application has visibility over all the IoT nodes and can process all the analytics needed today and in the future.

However, 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.

An important design consideration, therefore, is how to design an IoT network to manage this volume of data in an efficient way such that the data can be quickly analyzed and lead to business benefits. The volume of data generated by IoT devices can be so great that it can easily overrun the capabilities of the headend system in the data center or the cloud. For example, it has been observed that a moderately sized smart meter network of 1 million meters

will generate close to 1 billion data points each day (including meter reads and other instrumentation data), resulting in 1 TB of data. For an IT organization that is not prepared to contend with this volume of data storage and real-time analysis, this creates a whole new challenge.

The volume of data also introduces questions about bandwidth management. As the massive amount of IoT data begins to funnel into the data center, does the network have the capacity to sustain this volume of traffic? Does the application server have the ability to ingest, store, and analyze the vast quantity of data that is coming in? This is sometimes referred to as the "impedance mismatch" of the data generated by the IoT system and the management application's ability to deal with that data.

As illustrated in Figure 2-14, data management in traditional IT systems is very simple. The endpoints (laptops, printers, IP phones, and so on) communicate over an IP core network to servers in the data center or cloud. Data is generally stored in the data center, and the physical links from access to core are typically high bandwidth, meaning access to IT data is quick.

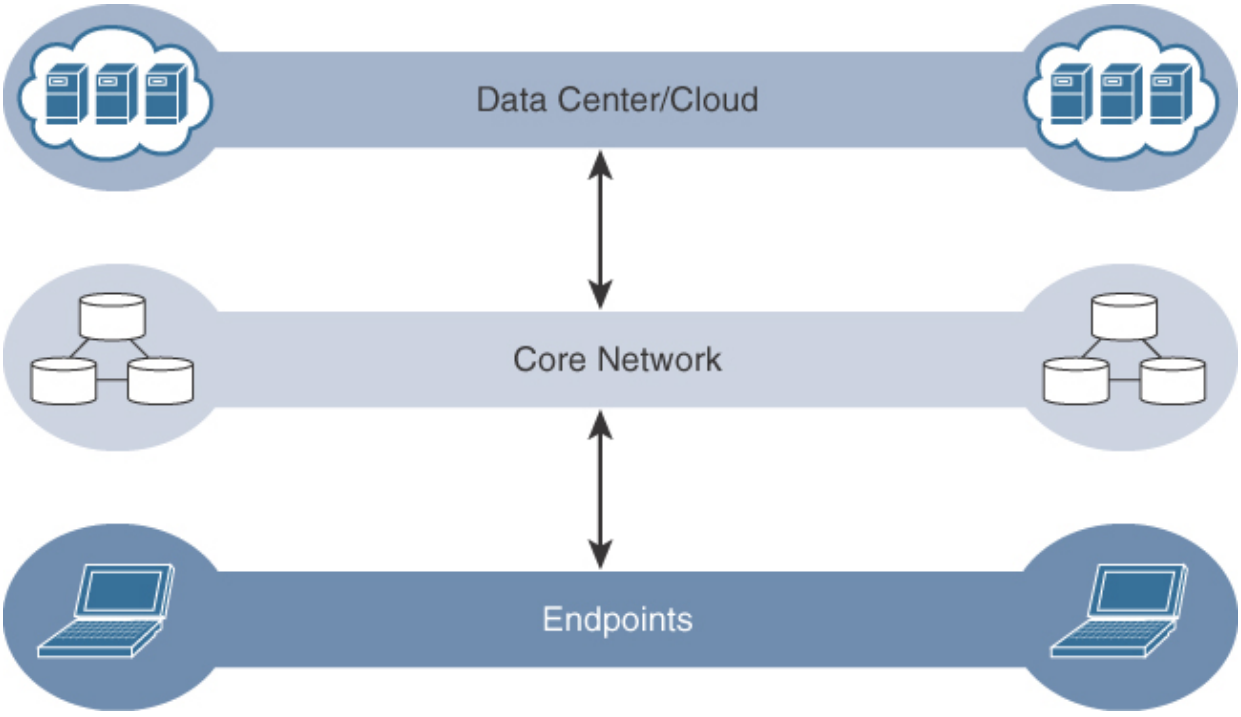

Figure 2-14 *The Traditional IT Cloud Computing Model*

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.

Note

The concept of fog was first developed by Flavio Bonomi and Rodolfo Milito of Cisco Systems. In the world of IoT, fog gets its name from a relative comparison to computing in the cloud layer. Just as clouds exist in the sky, fog rests near the ground. In the same way, the intention of fog computing is to place resources as close to the ground—that is, the IoT devices—as possible. An interesting side note is that the term "fog" was actually coined by Ginny Nichols, Rodolfo's wife. Although not working directly in IoT, she had an excellent grasp of what her husband was developing and was able to quickly draw the comparison between cloud and edge computing. One day she made the suggestion of simply calling it the "fog layer." The name stuck.

An advantage of this structure is that the fog node allows intelligence gathering (such as analytics) and control from the closest possible point, and in doing so, it allows better performance over constrained networks. In one sense, this introduces a new layer to the traditional IT computing model, one that is often referred to as the "fog layer." Figure 2-15 shows the placement of the fog layer in the IoT Data Management and Compute Stack.

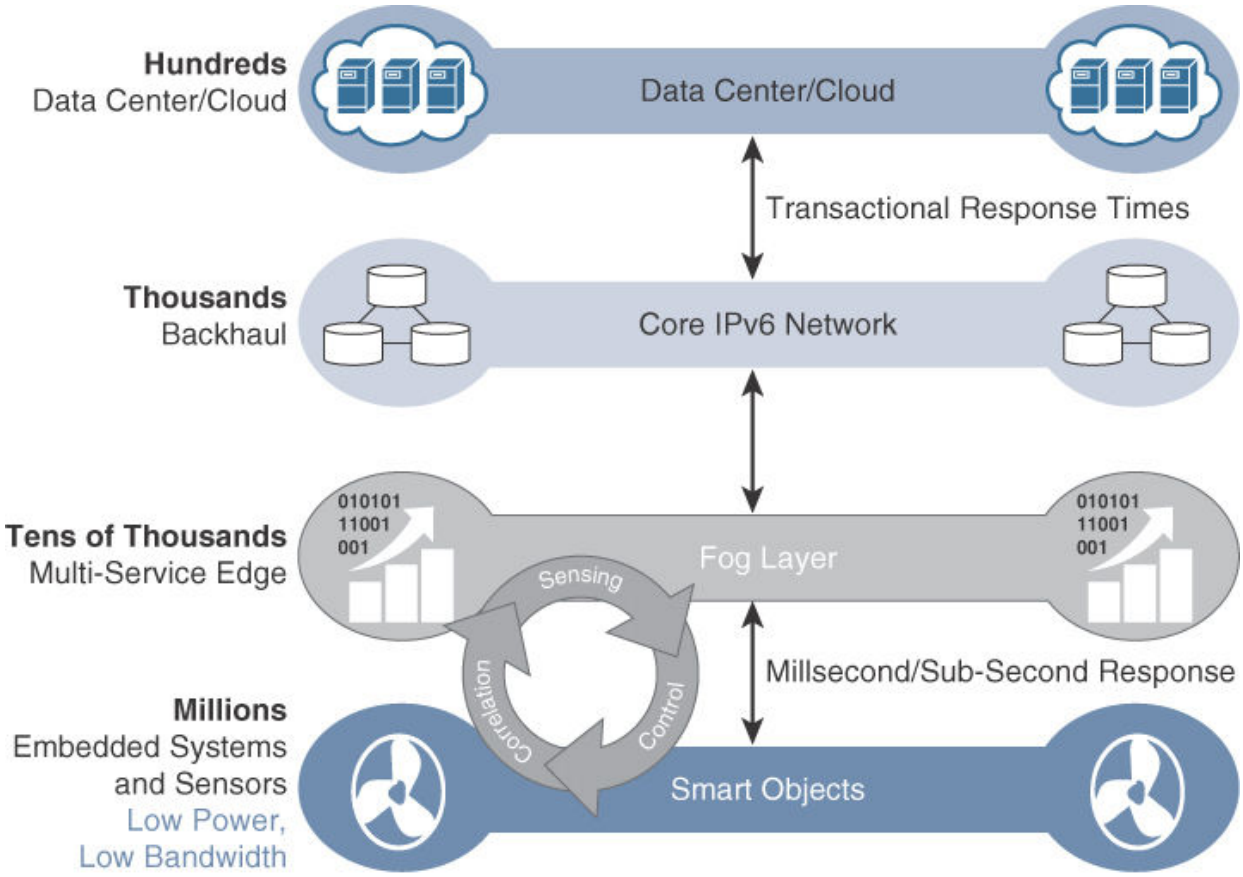

Figure 2-15 *The IoT Data Management and Compute Stack with Fog Computing*

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.

In addition, having contextual awareness gives fog nodes the ability to react to events in the IoT network much more quickly than in the traditional IT compute model, which would likely incur greater latency and have slower response times. The fog layer thus provides a distributed edge control loop capability, where devices can be monitored, controlled, and analyzed in real

time without the need to wait for communication from the central analytics and application servers in the cloud.

The value of this model is clear. For example, tire pressure sensors on a large truck in an open-pit mine might continually report measurements all day long. There may be only minor pressure changes that are well within tolerance limits, making continual reporting to the cloud unnecessary. Is it really useful to continually send such data back to the cloud over a potentially expensive backhaul connection? With a fog node on the truck, it is possible to not only measure the pressure of all tires at once but also combine this data with information coming from other sensors in the engine, hydraulics, and so on. With this approach, the fog node sends alert data upstream only if an actual problem is beginning to occur on the truck that affects operational efficiency.

IoT fog computing enables data to be preprocessed and correlated with other inputs to produce relevant information. This data can then be used as real-time, actionable knowledge by IoT-enabled applications. Longer term, this data can be used to gain a deeper understanding of network behavior and systems for the purpose of developing proactive policies, processes, and responses.

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. However, in recent years, the concept of IoT computing has been pushed even further to the edge, and in some cases it now resides directly in the sensors and IoT devices.

> **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.

IoT devices and sensors often have constrained resources, however, as compute capabilities increase. Some new classes of IoT endpoints have enough compute capabilities to perform at least low-level analytics and filtering to make basic decisions. For example, consider a water sensor on a fire hydrant. While a fog node sitting on an electrical pole in the distribution network may have an excellent view of all the fire hydrants in a local neighborhood, a node on each hydrant would have clear view of a water pressure drop on its own line and would be able to quickly generate an alert of a localized problem. The fog node, on the other hand, would have a wider view and would be able to ascertain whether the problem was more than just localized but was affecting the entire area. Another example is in the use of

smart meters. Edge compute–capable meters are able to communicate with each other to share information on small subsets of the electrical distribution grid to monitor localized power quality and consumption, and they can inform a fog node of events that may pertain to only tiny sections of the grid. Models such as these help ensure the highest quality of power delivery to customers.

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.

This model suggests a hierarchical organization of network, compute, and data storage resources. At each stage, data is collected, analyzed, and responded to when necessary, according to the capabilities of the resources at each layer. As data needs to be sent to the cloud, the latency becomes higher. The advantage of this hierarchy is that a response to events from resources close to the end device is fast and can result in immediate benefits, while still having deeper compute resources available in the cloud when necessary.

It is important to note that the heterogeneity of IoT devices also means a heterogeneity of edge and fog computing resources. While cloud resources are expected to be homogenous, it is fair to expect that in many cases both edge and fog resources will use different operating systems, have different CPU and data storage capabilities, and have different energy consumption profiles. Edge and fog thus require an abstraction layer that allows applications to communicate with one another. The abstraction layer exposes a common set of APIs for monitoring, provisioning, and controlling the physical resources in a standardized way. The abstraction layer also requires a mechanism to support virtualization, with the ability to run multiple operating systems or service containers on physical devices to support multitenancy and application consistency across the IoT system. Definition of a common communications services framework is being addressed by groups such as oneM2M, discussed earlier. Figure 2-16 illustrates the hierarchical nature of edge, fog, and cloud computing across an IoT system.

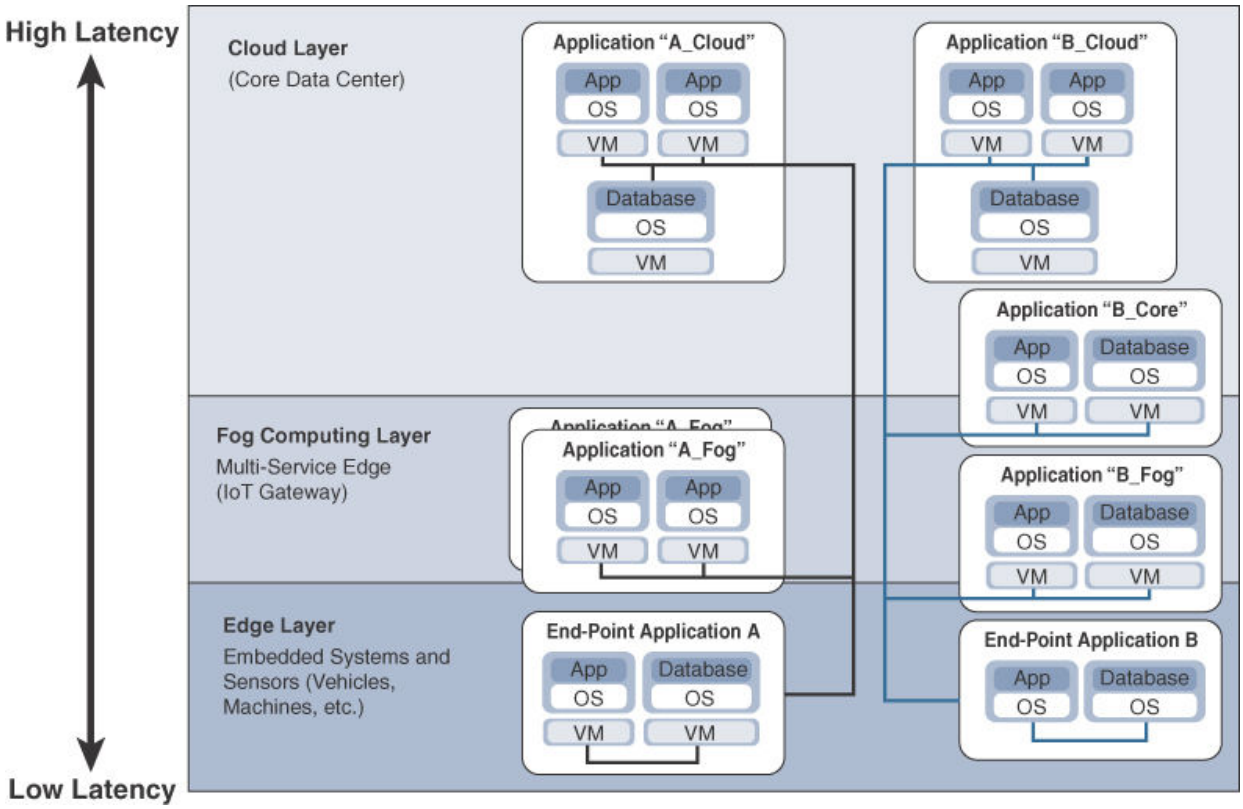

Figure 2-16 *Distributed Compute and Data Management Across an IoT System*

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.

Summary

The requirements of IoT systems are driving new architectures that address the scale, constraints, and data management aspects of IoT. To address these needs, several IoT-specific reference models have arisen, including the oneM2M IoT model and the IoT World Forum's IoT Reference Model. The commonalities between these models are the interaction of IoT devices, the network that connects them, and the applications that manage the endpoints.

This book presents an IoT framework that uses aspects of these various models and applies them to specific industry use cases. This chapter presents a model based on common concepts in these architectures that breaks the IoT layers into a simplified architecture incorporating two parallel stacks: the Core IoT Functional Stack and the IoT Data Management and Compute Stack. This architecture sets the format for the chapters that follow in this book.

The Core IoT Functional Stack has three layers: the IoT sensors and actuators, networking components, and applications and analytics layers. The networking components and applications layers involve several sublayers corresponding to different parts of the overall IoT system.

The IoT Data Management and Compute Stack deals with how and where data is filtered, aggregated, stored, and analyzed. In traditional IT models, this occurs in the cloud or the data center. However, due to the unique requirements of IoT, data management is distributed as close to the edge as possible, including the edge and fog layers.

References

1. ETSI, *oneM2M*, Accessed December 2016, www.etsi.org/about/what-we-do/global-collaboration/onem2m.
2. oneM2M, *oneM2M Technical Specification*, Accessed December 2016, ftp.onem2m.org/Deliverables/20140801_Candidate%20Release/TS-0002-Requirements-V-2014-08.pdf.