Part II

Enablers

3

An Overview of Enabling Technologies for the Internet of Things

Faisal Alsubaei,[1] Abdullah Abuhussein,[2] and Sajjan Shiva[1]

[1]*The University of Memphis, Memphis, TN, USA*
[2]*St. Cloud State University, St. Cloud, MN, USA*

3.1 Introduction

The Internet of Things (IoT) initially utilized current Internet infrastructure and existing technologies to transform stand-alone objects (i.e., devices) into interconnected smart objects. Before the development of IoT-specific technologies, computers and networking technologies were applied, with major tweaks, to support IoT applications. For example, IPv4 that was used in wireless sensor networks (WSNs) was used extensively in the beginning of IoT to connect nodes to the Internet using one IP address (i.e., Network Address Translation (NAT)), which requires extra effort to setup. However, many issues emerged from the use of technologies that were not designed with IoT in mind (e.g., the lack of autoconfiguration in IPv4). As such, there was a need to develop new technologies to overcome various issues, such as efficient routing, scalability, and mobility, to make developing IoT applications easier and more efficient. These advancements in IoT technologies have resulted in extensive real-world deployment of IoT applications in many domains, such as healthcare, industry, and smart buildings.

Considering the development of IoT enabling technologies during the past decade, it is clear that the most distinct trends are related to efficiency, which requires the optimal use of resources (e.g., battery and memory). Low-power consumption is an important factor for designing new IoT applications. Devices save energy while they are in "sleep" mode and consume energy when they are active (or awake). Devices are designed to wake up based on an optimum schedule to perform some operations (i.e., collecting and sending data). The complexity of these operations must be optimized to increase the lifespan of devices. Designers strive to achieve low device complexity, low energy

Internet of Things A to Z: Technologies and Applications, First Edition. Edited by Qusay F. Hassan.

consumption (i.e., long network lifetime), and an appropriate balance between signal/data processing capabilities and communication. An important issue in efficiency is the lack of interoperability in IoT (i.e., devices from different brands and models cannot work well together without a translation layer). The interoperability issue remains a major challenge in current IoT technologies because of the lack of unified standards for the IoT. Hopefully, as the IoT matures, these problems will be minimized.

This chapter details the key concepts of the enabling technologies based on their positions in the well-known five-layer model, which consists of the perception layer, network layer, middleware layer, application layer, and business layer.

3.2 Overview of IoT Architecture

Managing millions of heterogeneous connected devices via the Internet requires a flexible, layered architecture. As introduced in Chapter 1, there are various IoT architectural models, namely, Internet of Things — Architecture (IoT-A), Industrial Internet Reference Architecture (IIRA), Reference Architecture Model Industrie 4.0 (RAMI 4.0), and Cisco's Internet of Things Reference Model. However, no standard reference architecture has been adopted (Al-Fuqaha et al., 2015; Al-Qaseemi et al., 2016).

There are also ongoing projects that aim at designing new architectures for IoT such as P2413 - Standard for an Architectural Framework for the IoT[1], European Research Cluster on the Internet of Things (IREC)[2], Internet of Things Reference Architecture (IoT RA)[3], and Arrowhead Framework[4]. Additionally, there are other architectures that are related to the IoT but only focus on subsets of the IoT. For example, the European Telecommunications Standards Institute Technical Committee (ETSI TC) for M2M[5] is focused only on the communication standards while the International Telecommunication Union Telecommunication Standardization Sector (ITU-T)[6] focuses only on the identification systems (Weyrich and Ebert, 2016).

In order to systematically discuss the IoT enabling technologies, they will be discussed in this chapter based on their appearance on an architecture model. For the purpose of this chapter (i.e., to provide an overview of IoT generic technologies), none of the above reference architectures are suitable for the

1 http://standards.ieee.org/develop/project/2413.html
2 http://www.internet-of-things-research.eu/
3 https://www.iso.org/standard/65695.html
4 http://www.arrowhead.eu/
5 http://www.etsi.org/technologies-clusters/technologies/internet-of-things
6 http://www.itu.int/en/ITU-T/Pages/default.aspx

following reasons. First, IoT-A is relatively old and does not define IoT key concepts like cloud and fog computing, which are left to be identified in the implementation (Weyrich and Ebert, 2016). Second, IoT-A focuses only on the WSNs and the application activities (Gubbi et al., 2013; Li et al., 2015) and does not include other technologies that are important in modern IoT applications (such as processing Big Data for analytics). Third, RAMI 4.0 and IIRA are mostly for Industrial IoT (IIoT) and do not expand to other IoT domains such as Internet of Medical Things (IoMT), Internet of Vehicles (IoV), and so on. Moreover, Cisco's reference model cannot be used because it has not been finalized yet. Finally, because technologies will be presented without a specific IoT scenario in mind, a pseudophysical viewpoint is appropriate; however, none of the above architectures provides this viewpoint.

Therefore, this chapter categorizes the IoT enabling technologies based on a comprehensive, generic, and simple architecture, namely, the IoT five-layer model as depicted in Figure 3.1. This model captures many of the essences of other models (and simplifies and systematizes them). Hence, this architecture has been used extensively in recent publications (Al-Fuqaha et al., 2015; Al-Qaseemi et al., 2016). In Figure 3.1, each layer includes examples of technologies that are categorized by their functionality. These layers are conceptually similar to the Transmission Control (TCP)/Internet Protocol (IP) layers, as will be discussed in detail in the next section. Although many IoT technologies can fit into the TCP/IP (or Open Systems Interconnection (OSI)) model, categorizing all IoT technologies based on the TCP/IP model does not encompass all technologies. Thus, their classification based on the five-layer model is more appropriate.

3.3 Enabling Technologies

IoT enabling technologies vary depending on the domain and scenario. For example, smart transportation requires flexible technologies that ensure the connectivity of a vast number of mobile nodes. In contrast, the focus in healthcare is reliability and integrity. Therefore, generic technologies are briefly presented here based on their functionality in the IoT five-layer model.

3.3.1 Perception Layer Technologies

The perception layer (also known as the objects layer) is the first layer (from the bottom, as shown in Figure 3.1) in the IoT model. It includes various types of physical devices that are responsible for collecting data and acting accordingly, such as object identifiers, temperatures, locations, and humidity measurements. The power consumption and communication ability (i.e., unidirectional or

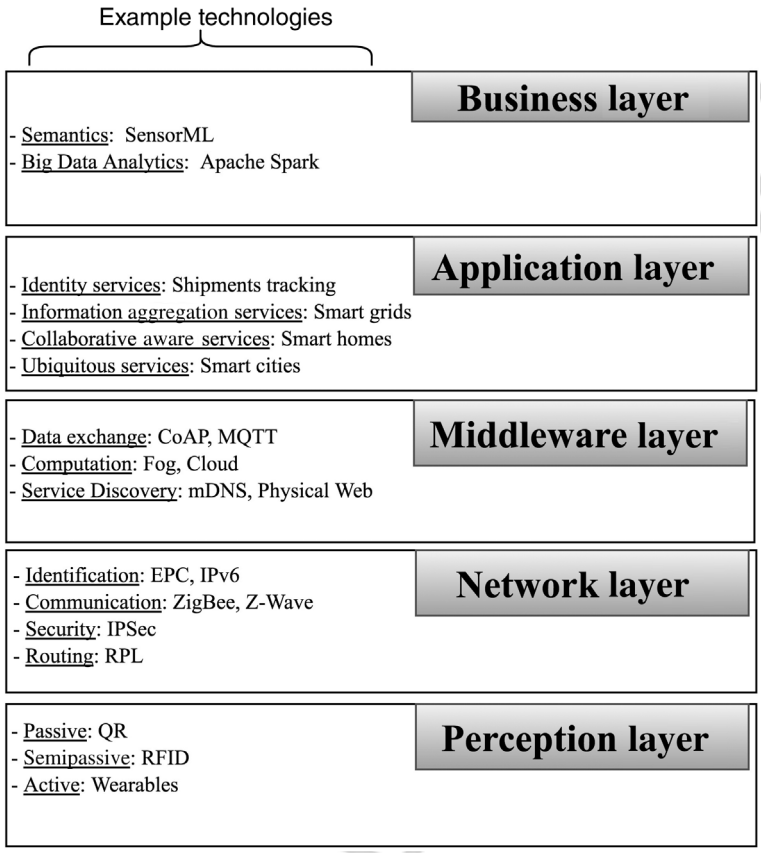

Figure 3.1 The IoT five-layer architectural model. (Reproduced with permission from IEEE Press.)

bidirectional) are important aspects of this layer. The classification of the perception layer technologies is as follows:

3.3.1.1 Passive

Passive devices have the lowest radio coverage because they do not have onboard power supplies and rely on readers in close proximity for their energy supply. The limitation of passive enabling technologies is that communication with them is unidirectional. Data can be read from nodes but cannot be written back to the nodes. However, for many applications, unidirectional communication is sufficient. Examples of passive perception layer technologies are as follows:

- *Quick Response (QR) Codes:* These codes function as a method of storing information on an IoT node and can be used to recognize nodes. QR codes are

probably the easiest and cheapest technology in this category. They can potentially be applied to almost anything. By printing a 1 sq. in. code on nodes, they become readily recognizable using various readers, such as cameras or light sensors, although they must appear within the line of sight of the reader. QR codes can be employed in many scenarios, such as shipments or manufacturing tracking.

- *Passive Radio Frequency Identification (RFID) Tags:* These tags are small electronic chips that — similar to QR codes — can be attached to any node enabling the nodes to be recognized for tracking or automatic identification purposes, which enable applications such as inventory management. Readers recognize these chips when they are within a short range by sending a signal that provides sufficient power for the tag to respond; the response contains the identification data.

3.3.1.2 Semipassive

Semipassive devices have batteries that power tags while receiving signals from a reader. Unlike passive technologies, the extra available power enables them to read and write data. Examples of semipassive technologies are as follows:

- *Semipassive RFID Tags:* These tags are similar to passive tags, but their batteries enable them to have a longer range, which supports a variety of additional uses. One example is smart transportation, for example, blocking railway crossings when a train is approaching.
- *Infrared (IR) Readers:* These readers are devices that sense electromagnetic radiation that is invisible to humans. Modern uses of this technology include automatic and remote body temperature sensing and remote controls. In the context of IoT, connected IR readers can trigger some actions (e.g., alarm) when detecting abnormalities (e.g., high body temperature). Since the IR is cheap, simple, and consumes low energy, it might be preferable in some IoT scenarios where IR readers might be stolen (e.g., monitoring body temperature at airports). However, it is limited to the line of sight and is subject to interference or failure due to objects blocking the light signals.

3.3.1.3 Active

Active devices often have the longest range because they have larger batteries or are attached to power supplies that enable them to receive and transmit data. Examples of these devices are as follows:

- *Active RFID Tags:* Similar to semipassive RFID tags but have a longer range due to their larger onboard battery. These tags are mostly used in real-time locating systems (RTLS)[7], for example, to monitor valuable assets (e.g., to

7 Real-time location system (RTLS) is a set of technologies to track targets (e.g., vehicles, items in a factory, or people) based on their recent geolocation.

track lost or stolen expensive medical devices). In such IoT scenarios, connected nodes equipped with active RFID can trigger an alarm when they leave a particular area.

- *Smart Actuators:* Mechanical devices that can be automatically controlled, without human intervention, to apply a specific motion to a product or process based on sensed data or based on the defined workflow. Connected actuators can help industries or smart cities to be more efficient and save money because actuators can react automatically based on sensed data, which requires less staff.
- *Wearable, Embedded, or Stand-Alone Smart Sensors:* Devices that include embedded sensors, such as gyroscopes, Global Positioning System (GPS) radios, and heart rate monitors used in modern smartphones or smart watches to monitor users' activities and health. Smart thermostats also use connected sensors to efficiently manage energy. In modern IoT applications, such connected devices (e.g., smartwatch, smartphone, etc.) often collaborate to achieve a common goal (e.g., pervasive health monitoring, alert care givers, etc.).

3.3.2 Network Layer Technologies

The network or object abstraction layer is the second layer. It is considered the infrastructure layer because its technologies transform the conventional sensors described in the perception layer into smart and connected nodes. Network layer technologies enable nodes to be identifiable over the Internet or any local area network, which allows them to securely communicate with each other. Many technologies in this layer are also found in the first three layers in the IP suite (TCP/IP) (i.e., Link, Internet, and Transport layers). Due to the restricted capabilities (i.e., power and computing) of most IoT nodes, scalable and efficient routing techniques are required to ensure interoperability among IoT devices. Note that many of IoT technologies are actually utilized in WSNs or machine-to-machine (M2M) communications but were improved to meet IoT requirements (i.e., Internet connectivity). Many modern devices use more than one technology; as in smart watches, which often have Wi-Fi, Bluetooth, and NFC. In this section, we categorize the enabling technologies based on their functionality in the network layer.

3.3.2.1 Identification

When a node connects to the network, it is assigned an identifier that enables it to communicate with other nodes. Identification is important to control the excessive use of bandwidth in large networks (e.g., flooding) by ensuring that only identified nodes communicate. To easily locate devices in the massive network, names and addresses are assigned to the devices. The naming

conventions assign structured names to nodes; these names allow the nodes to be readily identified and may even specifically describe the functions. Naming solutions are as follows:

- *Uniform Resource Identifier (URI)*[8]*:* A unique sequence of characters in the form of a web link that refer to an abstract or physical resource (Masinter et al., 2016). These identifiers are used to locate and interact with resources (including IoT nodes) on networks using human-friendly structured names that describe their purpose (e.g., http://www.example.com/temperature/actual).
- *Electronic Product Code (EPC)*[9]: An EPC global tag standard that assigns a unique identity to each physical object anytime worldwide (e.g., EPC = urn:epc:id:sgtin:0614141.012345.62852). EPCs are encoded on passive RFID tags, which can be used to track all types of objects, such as vehicles on toll roads.
- *Ubiquitous Code (uCode)*[10]: A numeric identification system that uniquely identifies physical objects and places in the real world. Using uCode, information can be associated with objects and locations and can be retrieved from www.uidcenter.org. An example uCode is 00001C00000000000001000285E7A6E3. Unlike EPC, uCode offers more flexibility because the tags can take many forms, such as print (i.e., QR and barcodes), passive RFID, active RFID, infrared, and acoustic tags.

Addressing techniques enable the assignment of unique network addresses to nodes for better network management. Addressing protocols are as follows:

- *IPv4*: A popular networking protocol that uses 32-bit addresses to assign nodes to unique addresses and aggregate them into smaller networks. Although IPv4 is extensively applied, it is not recommended for IoT due to the small number of addresses (i.e., 2^{32} addresses) that can be employed.
- *IPv6*: A networking protocol that aims to overcome the limitations of IPv4 by supporting 2^{128} addresses, which allows every device to have a unique address. Unlike IPv4, IPv6 provides autoconfiguration, which enables devices to be assigned IPs without the need for a Dynamic Host Configuration Protocol (DHCP) server. Autoconfiguration is an important feature for IoT because it simplifies the large-scale deployment of WSNs. Due to the large header size (i.e., 320 bits) in IPv6, *IPv6 over Low Power Wireless Personal Area Networks* (6LoWPAN) is used to compress the address to make it compatible with traditional WSN protocols (i.e., IEEE 802.15.4), as will be subsequently described.

8 https://tools.ietf.org/html/rfc3986
9 www.gs1.org/epc-rfid
10 www.uidcenter.org

3.3.2.2 Communication

After a node has been identified (i.e., addressed and named), it can start communicating with other nodes or with backend servers. However, this communication requires the selection of a suitable communication medium, which depends on the capabilities of the node (e.g., power and coverage). Nodes can communicate horizontally (i.e., ad hoc with another node) or vertically (i.e., with servers in the middleware layer). Communication technologies are discussed in this section with emphasis on their use in the IoT. These commonly employed technologies are listed in ascending order based on their wireless range (i.e., wired and short, medium, and long range):

- *Power-Line Communication (PLC):* This comprises a set of communication protocols that use power-line wiring to simultaneously transmit both data and alternating current. Using this approach, a person can power devices and control/retrieve data using only the standard power cables that run to the device. PLC can be used to connect the nodes in an IoT environment to share data with the backend that then performs some actions. It is mainly preferable in stationary nodes because they rely on power lines anyway as well as in IoT environments that have high wireless interference in smart vehicles, smart grids, and smart homes.
- *X10:* Similar to PLC, X10 is an industry standard that uses electrical cabling for signaling and controlling devices, in which the signals contain short Radio Frequency (RF) bursts that can include data. Despite its lower data rate and range compared with PLC, it remains a popular choice due to its low cost. X10 is primarily employed to connect nodes in smart homes where outlet modules (e.g., light switches, AC switches, etc.) can be controlled from a mobile application or a web portal.
- *Near-Field Communication (NFC)*[11]: A set of protocols that enable communication between two devices over very short distances. NFC is unlike RFID, which is used primarily for identification, it offers a simple way to authenticate, access, and share data between devices and users in IoT systems. NFC is employed in many IoT applications such as smart payments and access control systems.
- *Ultra-Wide Bandwidth (UWB):* An older wireless technology that uses low-energy transmissions to provide high-bandwidth communications over a wide radio spectrum (Fontana, 2004). This technology is suitable for IoT due to its resistance to interference, which makes it highly scalable since adding more nodes will not cause interference. This technology can help wireless networks to be more adaptive and efficient because each node can be tracked using the

11 www.nfc-forum.org

UWB. UWB is usually employed in location-based services (e.g., asset tracking) (Anonymous, 2017).

- *Wi-Fi:* Wi-Fi is particularly useful for ad hoc configurations, such as Wi-Fi Direct, which does not require a wireless access point. The main limitation of Wi-Fi is its power consumption. However, in some IoT applications (e.g., smart homes), power is not an important issue. The Wi-Fi Alliance is launching a new energy-efficient Wi-Fi technology named Wi-Fi HaLow[12] that is specifically designed for IoT nodes.
- *IEEE 802.15.4*[13]: A standard for low-rate wireless personal area networks (LR-WPANs) that specifies the physical layer and media access control. Because IEEE 802.15.4 was designed for IPv4, the larger sized datagrams of IPv6 are not a natural fit for IEEE 802.15.4 networks. The solution to this problem is to use an adaptation layer (i.e., 6LoWPAN), which encapsulates and compresses headers to allow IPv6 packets to be carried over IEEE 802.15.4 networks. ZigBee[14], ISA100.11a[15], WirelessHART[16], MiWi[17], and Thread[18] are based on IEEE 802.15.4 and only differ in their upper layers. ZigBee and Thread are commonly employed in smart homes and smart meters, whereas ISA100.11a, MiWi, and WirelessHART are commonly used in industrial applications.
- *Bluetooth Low Energy (BLE)*[19]: A wireless standard is intended to exchange data over short distances and build *personal area networks* (PANs). For Bluetooth v4.0 and later versions, the energy consumption is improved, which renders Bluetooth well suited for sensors and other small devices that require low power. Internet-enabled nodes often use BLE to act with local nodes and send collected data to the backend for more actions. BLE can be used in an extensive range of applications, such as smart buildings, smart transportation, and wearables.
- *ANT+*[20]: A proprietary (but open access) wireless communications protocol stack enables nodes to communicate by creating rules for coexistence, signaling, data representation, authentication, and error detection. It is primarily employed in sports, wellness management, and home health monitoring IoT applications where each IoT node use or send sensed data to the cloud, for more actions.

12 www.wi-fi.org/discover-wi-fi/wi-fi-halow
13 www.standards.ieee.org/about/get/802/802.15.html
14 www.zigbee.org/
15 www.isa.org/isa100/
16 www.fieldcommgroup.org/technologies/hart/hart-technology
17 www.microchip.com/design-centers/wireless-connectivity/embedded-wireless/802-15-4/miwi-protocol
18 www.threadgroup.org
19 www.bluetooth.com/what-is-bluetooth-technology/how-it-works/low-energy
20 www.thisisant.com/developer/ant-plus/

- *Z-Wave*[21]: A technology extensively applied in smart homes. Z-Wave devices can be attached to home appliances, which enable them to be controlled over the Internet.
- *Weightless*[22]: A set of Low-Power Wide-Area Network (LPWAN)[23] open wireless technology standards for exchanging data between a base station and thousands of surrounding nodes. There are three different standards for Weightless, namely, Weightless-W, Weightless-P, and Weightless-N; each of which has different use cases due to its features. Weightless-W offers two-way communication that is ideal for use in the smart oil and gas sector, where TV white space (TVWS) is likely to be available. Although weightless-P also provides two-way communication, it is perfect for private networks and situations that require both uplink and downlink capabilities. Weightless-N offers an uplink that makes it ideal for sensor-based networks, such as temperature readings, tank level monitoring, and metering.
- *Cellular Networks:* These networks exist in various generations (e.g., 3G and 4G) of cellular network standards that are often employed in smartphones but are also suitable for IoT due to their high mobility and higher speed. Substantial progress has been made in recent generations in terms of power consumption and speed. For example, a recent specification of 4G (called LTE-Advanced) offers a power saving mode that extends the device battery life to a decade with a maximum speed of 10 Mbps. For comparison, other versions of 4G (less energy efficient) can have a speed of up to 1 Gbps (Nokia, 2017).
- *SigFox*[24]*:* A subscription-based service that offers connectivity solutions (in some countries) over dedicated LPWAN networks. SigFox is ideal for IoT applications that need to send infrequent and small bursts of data, such as alarm systems or smart meters.
- *DASH7 Alliance Protocol (D7A)*[25]: An open-source wireless sensor and actuator network protocol that provides multiyear battery life and low latency to connect moving nodes. Its IoT applications include smart buildings, location-based services, logistics, and smart vehicles.
- *Long-Range Wide-Area Network (LoRaWAN)*[26]: A technology from LoRa Alliance that offers low-cost, mobile, and secure bidirectional communication. It is optimized for low power consumption and is designed to support large networks with millions of nodes, which renders it ideal for IoT applications such as smart cities and industrial applications.

21 www.z-wave.com
22 www.weightless.org
23 LPWAN is a form of wireless network that allows connected objects (things) to communicate over a wide coverage area at a low speed.
24 www.sigfox.com
25 www.dash7-alliance.org
26 www.lora-alliance.org/For-Developers/LoRaWANDevelopers

Some additional technologies that are expected to be adopted in IoT applications are currently under development and will likely be used in the next few years because they will address some of the communication limitations in current technologies. This includes, for example, the following:

- *5G:* A medium-range communication candidate that should be available by 2020 (Prasad, 2014). It offers many improvements over 4G, such as increased speed (almost equivalent to Ethernet) and better coverage, and supports a large number of users. Thus, it will enable a larger number of nodes to be connected with extra mobility.
- *Light Fidelity (Li-Fi):* A new wireless technology that offers high-speed data transmission using visible light communication (VLC) (Ackerman, 2015). Li-Fi can also utilize energy-efficient light-emitting diode (LED) lights to reduce energy costs. However, the Li-Fi scheme is limited to the line of sight and is subject to interference or failure due to objects blocking the light signals. Although Li-Fi has not been extensively adopted in IoT, its high speed indicates that it has substantial potential for use in many applications, such as smart theaters and smart classrooms.
- *Software Defined Network (SDN):* A networking architecture that is dynamic, manageable, adaptable, and cost effective. These characteristics make SDNs very effective, given the high-bandwidth and dynamic nature of the IoT (Jacquenet and Boucadair, 2016). Underutilized network resources and intelligent route trafficking will facilitate planning for the data flood that is expected as IoT usage increases. SDNs minimize bottlenecks and introduce network efficiencies (Duffy, 2014).

Each of these current communication technologies has special features that enable different scenarios to be implemented efficiently. Table 3.1 compares the important characteristics of most IoT communication technologies and lists them in ascending wireless range order. The data rate aids in the selection of the best technology based on the data flow requirements of applications. Similarly, range is an important consideration for IoT nodes in terms of mobility. Security is also a vital feature because large-scale sensors can cause catastrophic damage when compromised. Therefore, technologies that do not offer any built-in security should only be employed with additional security protocols provided by other layers, as discussed in the next section. Ad hoc communications are critical in applications where nodes communicate with each other without a controller. Technologies that are considered native TCP/IP protocols can be easily integrated into current systems. Header size may affect the choice of technologies with low data rates. Latency can be important in some IoT applications (e.g., remote surgery (West, 2016)). The total latency often increases when a large number of sensors are deployed. Frequency is important to avoid signal interference or restrictions in some places. Scalability is also an important feature of the IoT. Technologies that are scalable often allow for addition of new

Table 3.1 Comparison of communication technologies.

Technology	Data rate	Range	Mobility	Security	Ad hoc	Native TCP/IP	Header size	Latency	Frequency	Scalable
X10	20 bps	~20 m	No	No	Yes	No	8 bytes	Low	120 KHz	No
PLC	10 Mbps	~ 9 km	No	No	Yes	Yes	133 bytes	Varies	Narrowband (3–500 KHz) Broadband (1.8–250 MHz)	Yes
NFC	106, 212, 424, and 848 kbps	~20 cm	Yes	LPI/D[a)]	Yes	No	4 bytes	Low	13.56 MHz	No
UWB	480 Mbps–1.6 Gbps	~10 m	Yes	LPI/D	Yes	Yes	40 bits	Low	3.1–10.6 GHz	Yes
Z-Wave	9.6, 40, and 100 kbps	~30 m	Yes	No	Yes	No	Varies	Low	Regional sub-GHz bands	Yes
Thread	40–250 kbps	~30 m	Yes	TLS 1.2	No	No	40 bytes	Low	Global 2.4 GHz	Yes
MiWi	250 kbps	~20–50 m	Yes	TLS 1.2	Yes	No	11 bytes	Low	Regional sub-GHz and global 2.4 GHz	Yes
BLE	1 Mbps	~50 m	Yes	AES 128-bit	Yes	No	2 bytes	Low	Global 2.4 GHz	No
ANT+	1 Mbps	~50 m	Yes	AES 128-bit	Yes	No	14 bytes	Low	Global 2.4 GHz	Yes
Wi-Fi	250 Mbps	~60 m	Yes	WPA2 with AES	Yes	Yes	2 bytes	Low	Global 2.4, 5.8 GHz	Yes
ZigBee	20, 40, and 250 kbps	~100 m	Yes	128-bit Encryption	Yes	No	15 bytes	Low	Regional sub-GHz and global 2.4 GHz	Yes
ISA100.11a	250 kbps	~200 m	Yes	AES 128-bit	Yes	No	Varies	Low	Global 2.4 GHz	Yes
WirelessHART	250 kbps	~200	Yes	AES-128 bit	Yes	No	21 bytes	Low	Global 2.4 GHz	Yes

Wi-Fi	11 Mbps, 54 Mbps, 600 Mbps, 1300 Mbps, and 6.9 Gbps	~200 m	Yes	WPA2 with AES	Yes	Yes	2,346 bytes	High	Global 2.4, 5.8 GHz	Yes
D7A	167 kbps	1–5 km	Yes	AES 128-bit	Yes	No	3–38 bytes	Low	433 MHz, 868 MHz, 915 MHz	Yes
Sigfox	100 bps up, 600 bps down	~15 km	Yes	No	No	No	Varies	High	Regional sub-GHz bands	Yes
LoRa	0.3–50 kbps	~13 km	Yes	128-bit encryption	Yes	No	Varies	Low	Regional sub-GHz bands	Yes
3G	144–400 kbps (while moving)	Vary	Yes	KASUMI[b)]	Yes	No	Varies	Medium	UMTS 850 MHz–2100 MHz	Yes
4G	Up to 1 Gbps	Vary	Yes	Enhanced SNOW 3G[c)]	Yes	No	Varies	Low	LTE bands	Yes
Weightless-W	1–10 Mbps	~5 km	Yes	128-bit encryption	Yes	No	10 bytes or more	High	TVWS	Yes

a) Low Probability of Intercept/Detect (LPI/D) is a set of wireless security techniques that allow devices to see but not to be seen by modern and capable intercept receivers.
b) KASUMI is a block cipher with a 128-bit key and 64-bit input and output.
c) SNOW 3G is a word-based synchronous stream cipher.

large number of nodes without affecting the performance, whereas nonscalable technologies are generally limited to fewer than 100 nodes per network.

3.3.2.3 Security

Due to a large number of nodes with limited capabilities, security is an important and challenging task because a successful attack is likely to cause excessive damage (e.g., Distributed Denial-of-Service (DDoS) attacks) (Stephen, 2016). As discussed in the communication technologies section, security may not be built in to the various communication technologies (e.g., Z-wave and Sigfox). Therefore, additional security mechanisms must be provided by different layers to reduce the likelihood of attacks. Attacks can occur in this layer due to insecure communication among nodes. Therefore, lightweight security mechanisms are required for secure communication. Some suggested security techniques are as follows:

- *Internet Protocol Security (IPsec)*[27]*:* A well-known network layer protocol that is employed with IPv6 for authentication and end-to-end encryption among nodes. Because IPsec is implemented in the network layer (in TCP/IP), it also serves the upper layers. A lightweight implementation of IPsec is possible, which renders it ideal for IoT nodes (Raza et al., 2011).
- *Transport Layer Security (TLS)*[28] *and Datagram TLS (DTLS)*[29]*:* These are well-known cryptographic protocols that are also employed with TCP protocols and user datagram protocols (UDPs), respectively, to provide secure communications.
- *IEEE 1888*[30]*:* This protocol is part of a family of standards for the Ubiquitous Green Community Control Network. IEEE 1888.3 prevents unauthorized access to resources and accidental data leaks while enhancing the confidentiality and integrity of transferred data.

3.3.2.4 Routing

Once a node knows a destination address, it must have an efficient method for routing data to that destination node. Efficient routing is critical in IoT environments because of the massive number of nodes that can be connected in an ad hoc manner. Due to the limited capabilities of IoT nodes, each node must efficiently learn about the optimal route. Therefore, a special routing protocol was introduced for these environments. *Routing Protocol for Low Power and Lossy Networks* (RPL)[31] is an IPv6 routing standard protocol for resource-limited devices. This protocol was proposed to support different types of links,

27 https://tools.ietf.org/html/rfc4301
28 https://tools.ietf.org/html/rfc5246
29 https://tools.ietf.org/html/rfc6347
30 https://standards.ieee.org/findstds/standard/1888-2014.html
31 https://tools.ietf.org/html/rfc6550

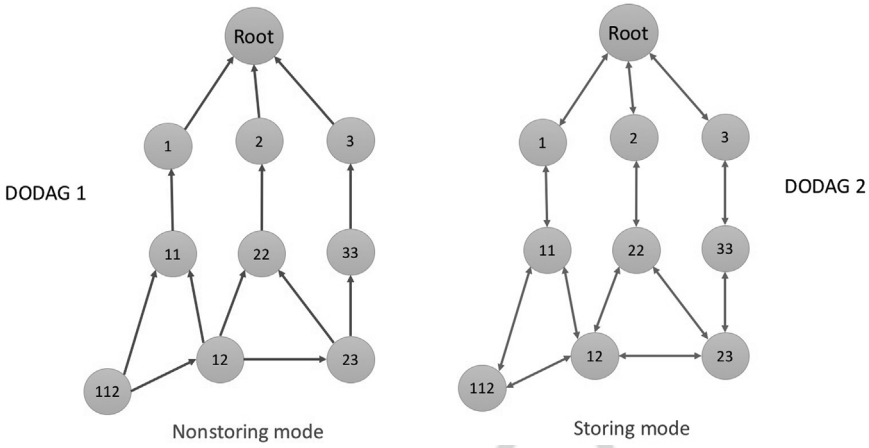

Figure 3.2 RPL topology.

such as IEEE 802.15.4, and common traffic types, including one-to-one, many-to-one, and one-to-many. The protocol can be compactly represented as a set of graphs in which each graph is a Destination-Oriented Directed Acyclic Graph (DODAG). In this type of graph, each node knows at least one path to its root node (i.e., border router), and each node is aware of its parent. Nodes exchange RPL special messages to maintain the graph and ensure that a valid route(s) to the root is always available. As shown in Figure 3.2, routers can work in two operational modes: nonstoring mode (as in DODAG 1) and storing mode (as in DODAG 2). In nonstoring mode, nodes forward each message to the root, which could be useful in scenarios where nodes are programmed to send sensed data (e.g., temperature readings) periodically. In contrast, in storing mode, communications are bidirectional, and each node stores the routes to all nodes under it. This can be helpful in scenarios where nodes only send sensed data when they are requested; hence, downward routes are needed to deliver the request messages.

3.3.3 Middleware Technologies

The middleware layer (also known as the service management layer) is the core of the IoT environment. It can be mapped to the application layer in the IP suite (TCP/IP). Technologies in this layer are often supported by IoT platforms. This layer enables services to be identified and requested based on names and addresses and enables programmers to interact with heterogeneous objects, regardless of the specific hardware setup. This layer also processes received data, makes decisions, and delivers required services (Al-Fuqaha et al., 2015). The

middleware layer performs those tasks using the following technologies, which are divided into three groups based on their functionalities.

3.3.3.1 Service Discovery

IoT applications and users need to be able to use names or addresses to request services without knowing the underlying infrastructure details. Therefore, services need to be discoverable and registered. Scalable and heterogeneous environments, service registration, and discovery should be autonomous, dynamic, and efficient. The protocols that are most commonly employed to provide service discovery are as follows:

- *Multicast DNS (mDNS)*[32]*:* A zero-configuration and infrastructure-independent protocol that can continue to function even during an infrastructure failure because they can use their local caches. It binds host names to IP addresses inside small networks by multicasting an IP request message asking a device with a particular host name to send its IP address. This host replies with a multicast message that contains its IP address. All other devices in the subnet use this response to update their mDNS local caches. This protocol, when combined with IPv6 (with autoconfiguration), can entirely eliminate the need for a local server to manage local WSNs.
- *DNS Service Discovery (DNS-SD)*[33]*:* Another zero-configuration protocol that complements mDNS because it pairs services to hosts within a domain. IoT clients can find a service by sending a multicast request for a service type (e.g., temperature readings), and all machines that provide this service will respond with their host names. Names are important because the nodes names, unlike IP addresses, rarely change. Then, mDNS is used to pair the names of the nodes with their IP addresses. Avahi[34] and Bonjour[35] are two popular implementations of both mDNS and DNS-SD.

  Another approach to service discovery is to publish the uniform resource identifiers (URIs) of nodes that provide services over the Web. This approach is known as the Web of Things (WoT). Technologies that adopt this approach are as follows:
- *HyperCat*[36]: An open, lightweight JavaScript Object Notation (JSON)-based hypermedia catalog format is intended to be used for publishing collections of URIs. Each HyperCat catalog can include any number of URIs, each with any number of Resource Description Format (RDF)[37]-like triple statements that

32 http://www.multicastdns.org/
33 http://www.dns-sd.org/
34 http://www.avahi.org/
35 https://support.apple.com/bonjour
36 www.hypercat.io
37 RDF is a World Wide Web Consortium (W3C) standard model for data interchange on the Web.

provide information about this URI. HyperCat enables developers to publish linked data descriptions of IoT nodes that can be fetched by HTTP GET requests, which requires IoT clients to know the URI address.

- *Physical Web*[38]: A technology that enables nodes to broadcast their Uniform Resource Locators (URLs) or some local data inside web pages. Unlike HyperCat, IoT clients do not need to know the URLs because they can fetch them via Bluetooth beacons (e.g., Eddystone[39]), mDNS, or Wi-Fi Direct (Anonymous, 2016a). The published URLs enable users to interact with the nodes (e.g., sensors) from their smartphones or laptops. A proxy service is used to prevent malicious URLs and sort lists of URLs based on users' preferences.
- *Universal Plug and Play (UPnP)*[40]: A group of networking protocols that enable networked IoT nodes to seamlessly discover each other's presence in the network and supply functional network services (i.e., data sharing, communications, and entertainment). However, UPnP is known to be vulnerable to security problems (Anonymous, 2016b). UPnP+ is an emerging initiative that aims to address the issues associated with UPnP (Anonymous, 2016c).

3.3.3.2 Data Exchange

To manage the vast amount of data created by the large number of nodes, a manager must transmit data to receivers in a secure and efficient manner. This management occurs on the TCP/IP application layer. Thus, the technologies that reside here are known as application protocols. In IoT environments, each node can capture data, analyze data, and control other nodes. However, before a node can begin communicating and exchanging data in the IoT network, it must be identified and registered as a part of the service network. This operation is referred to as node subscription. A subscribing node must make a request to join a service network (i.e., subscribe) before it can obtain or publish service-related data (e.g., temperature) from or to the network. Once the node is subscribed to publishers, it will receive service-related data when they are published. This model is known as the publish/subscribe model. An alternative approach is called request/response in which a joined node always requests a service-related data explicitly and receives a response that contains the requested data. Once the node joins the network, it is assigned network resources and can start exchanging data and operating as a component of the IoT environment. The publish/subscribe model is generally preferable for IoT scenarios that require the large deployment of nodes that frequently exchange data. Indeed, in such scenarios, using the request/response model leads to high network overhead, which can

38 https://google.github.io/physical-web/
39 https://developers.google.com/beacons/
40 https://openconnectivity.org/resources/specifications/upnp/specifications

cause increased power consumption. In contrast, the request/response model can be easily integrated into client/server applications, as will be discussed for the Constrained Application Protocol (CoAP). Some of the commonly used protocols are as follows:

- *CoAP*[41]*:* A request/response protocol was created by the *Constrained RESTful Environments* (CoRE) group for constrained devices. To provide resource-oriented interactions in a request/response architecture, it uses the HTTP commands (i.e., GET, POST, PUT, and DELETE) over UDP. Unlike TCP, UDP is suitable for lightweight IoT implementations. HTTP-CoAP proxies can be used to effortlessly convert interactions between HTTP and CoAP. To provide the reliability that UDP lacks, the header of CoAP messages has two bits that determine the required Quality of Service (QoS) level. CoAP includes four types of messages: *confirmable* (i.e., must be acknowledged by the receiver with an acknowledgment [ACK] packet), *nonconfirmable* (i.e., "fire and forget" messages that do not require acknowledgment), *acknowledgment* (i.e., an acknowledgment of a confirmable message), and *reset* (i.e., rejecting a message or removing an observer). CoAP has a simple back-off retransmission mechanism to detect and prevent duplicates for confirmable messages via a unique message ID, which is a 16-bit header field. DTLS can be used to provide security to CoAP applications; however, DTLS has problems because (1) its handshakes cause resource-limited nodes to consume additional resources, leading to reduced battery life; (2) it increases the conversion complexity between HTTP and CoAP; and (3) it prevents CoAP's multicast support.
- *Extensible Messaging and Presence Protocol (XMPP)*[42]*:* An IETF protocol for communicating over the Internet. It is appropriate for IoT because it provides near real-time, low-latency, and platform-independent communications in a decentralized manner. It runs over TCP and works in both publish/subscribe and request/response models. In XMPP, a client connects to a server using eXtensible Markup Language (XML) messages. The XML parsing creates high network overhead; however, methods are available to reduce this overhead (Waher and DOI, 2015). XMPP has built-in core TLS/Secure Sockets Layer (SSL) security features. QoS is not provided in XMPP; it inherits the TCP mechanisms for reliability, which are not suitable for the IoT (Karagiannis et al., 2015). To overcome this issue, a draft standard (i.e., XEP-0184: Message Delivery Receipts) has been proposed (Saint-Andre and Hildebrand, 2011).
- *Message Queue Telemetry Transport (MQTT)*[43]*:* A lightweight M2M communication protocol that utilizes the publish/subscribe model. It is suitable

41 https://tools.ietf.org/html/rfc7252
42 https://xmpp.org/
43 http://mqtt.org/

for slow, unreliable connections and computation-limited devices. MQTT includes three types of modules: *broker*, *publishers*, and *subscribers*. A subscriber can register for a specific service; then, the broker notifies interested subscribers when publishers start a service that interests them. MQTT runs on top of TCP and provides three levels of QoS: *at most once* (i.e., a message will be delivered once with no confirmation), *at least once* (i.e., a message will be delivered at least once, with confirmation required), and *exactly once* (i.e., a message will be delivered exactly once using a four-step handshake) (Karagiannis et al., 2015). A slightly different version of MQTT for sensor networks (i.e., MQTT-SN) is aimed at embedded devices on non-TCP/IP networks, such as ZigBee. Security is implemented by the MQTT broker using TLS/SSL, which requires subscriber authentication.

- *Advanced Message Queuing Protocol (AMQP)*[44]*:* An open standard for the IoT that provides message-oriented publish/subscribe communications capabilities. Similar to MQTT, message reliability is maintained using message delivery guarantees, such as *at most once, at least once*, and *exactly once*. TLS/SSL and/or Simple Authentication and Security Layer (SASL)[45] are used for security over TCP. Communications in AMQP are managed by two components: exchanges and message queues. Exchanges route messages to or among appropriate queues based on predefined rules. Message queues store messages in queues before they are sent to receivers. Although it lacks protocol support for Last-Value Queues (LVQs)[46], AMQP provides a richer group of messaging scenarios (e.g., one-to-one, broadcast, topic filtered, and request-reply) (StormMQ, 2017).
- *Data Distribution Service (DDS)*[47]*:* A M2M standard that was created by the Object Management Group (OMG). It is a data-centric publish/subscribe protocol that does not rely on a broker. Due to its multicast support, it is a very reliable protocol because it offers 23 different QoS policies. The DDS architecture consists of two interface levels. The first level — Data-Centric Publish-Subscribe (DCPS) — is responsible for information delivery to subscribers. The second level — the Data-Local Reconstruction Layer (DLRL) — is an optional layer that breaks data objects into separate elements to enable the simple integration of DDS into the application layer (Anonymous, 2016d). In addition to publishers and subscribers, a typical DDS application has the following elements: (1) *a domain* (e.g., medical devices), (2) *a topic* (i.e., a group of data from similar devices, such as electrocardiogram (ECG) readings from many devices), (3) an *instance*, (i.e., a target where data is

44 https://www.amqp.org/

45 https://tools.ietf.org/html/rfc4422

46 LVQs are special queues that only retain the last value of the messages (i.e., messages are discarded when a newer message with the same value is received).

47 http://www.omg.org/spec/DDS/

changing, such as "patient #13 ECG readings"), (4) *a sample* (i.e., a snapshot of an instance at a point in time), (5) *a Data-Writer* (i.e., a source of information about the topic), and (6) a *DataReader* (i.e., an observer of the topic).

Table 3.2 compares the most important features of data exchange protocols. The first two columns list the messaging model type (i.e., request/response or publish/subscribe). The Brokerless column indicates whether a broker between subscribers and publishers is required. Brokerless protocols do not rely on a broker to deliver messages, thereby eliminating the possibility of a single point of failure. The RESTful column indicates whether services are accessible via stateless operations, which is relevant in situations where communication is required upon request (as in HTTP). The Transport column shows the type of transport protocol that is employed in the technology. TCP is connection oriented, which offers high QoS but can be slow for large applications. The Multicast column specifies whether the technology enables multicast, which is preferable to reduce unwanted communications and add extra QoS. The Security column lists the security features supported by each technology. The QoS policies column specifies the number of QoS policies: Typically, a high number of policies is better. The Header Size column indicates the minimum size of a packet, which can be a consideration when the data rate is low, and communications are unreliable. The Discovery column specifies whether the protocol has discovery abilities in terms of services and node properties.

3.3.3.3 Computation

Computation is required to process the collected data and manage sensors. Computations in IoT environments can range from lightweight to complex operations. Lightweight computations are related to managing the platform, such as service access lists that involve encryption/decryption. Complex computations address the actual data (e.g., logic). Computations can be performed based on IoT application requirements in the following locations:

- *Local:* Local computations are performed using a processing unit or *system on a chip* (SoC). Applications may include networking and one-board sensing technologies, such as Arduino[48], UDOO[49], Intel's IoT products[50], and Raspberry PI[51]. This hardware is typically operated by real-time operating systems (RTOSs), such as Contiki OS[52] and RIOT OS[53]. This type can be used in small

48 https://www.arduino.cc/
49 http://www.udoo.org/
50 http://www.intel.com/content/www/us/en/internet-of-things/products-and-solutions.html
51 https://www.raspberrypi.org/
52 http://www.contiki-os.org/
53 https://www.riot-os.org/

Table 3.2 Comparison of IoT data exchange protocols.

Protocol	Request/ response	Publish/ subscribe	Brokerless	RESTful	Transport	Multicast	Security	QoS policies	Header size (Byte)	Discovery
CoAP	Yes	Yes	No	Yes	UDP	No	DTLS	4	4	Yes
MQTT MQTT-SN	No	Yes	No	No	TCP	Yes	SSL	3	2	No
XMPP	Yes	Yes	No	No	TCP	Yes	SSL	0	Varies	Yes
AMQP	No	Yes	No	No	TCP	No	SSL SASL	3	8	No
DDS	No	Yes	Yes	No	TCP UDP	Yes	SSL DTLS	23	Varies	Yes

IoT project (e.g., do-it-yourself (DIY) security systems (Anonymous, 2016e)) or in border routers for lightweight operations, such as filtering and compressing data, to reduce network overhead and load on servers in the cloud or fog.

- *Cloud:* Cloud computing is preferable for applications in which sensors that reside in different places send generated data to the cloud for centralized processing. This approach is flexible because cloud services can scale-up or scale-down on demand (Hassan et al., 2012). Cloud-based IoT platforms are the most common architecture in current real IoT deployments. Kaa[54] and DeviceHive[55] are examples of cloud-based IoT *Platform as a Service* (PaaS) implementations. Cloud *Infrastructure as a Service* (IaaS) is used by IoT service providers to allow their customers to use IoT services (i.e., *Software as a Service* (SaaS)).
- *Fog:* A fog computing layer is ideally employed with the cloud to improve performance because it can be deployed near end users or nodes. This is important because cloud servers might be far away from the nodes; thus, fog computing servers can perform some operations to reduce the latency of transferring data all the way to the cloud. Also, fog is preferred over cloud because only a small amount of the data transferred is relevant for actionable insights. Hence, cloud is not always efficient. Thousands of fog computing units are provided by mobile network operators but do not have the same computing capabilities as the cloud. Similar to cloud computing, fog computing improves scalability. Low latency and location awareness, widespread graphical distribution, device density, mobility, and real-time capabilities are advantages of fog computing (Saint-Andre and Hildebrand, 2011). For more details about fog computing in the IoT, refer to Chapter 4.

3.3.4 Application Layer Technologies

This layer is responsible for providing requested services to IoT users via a simple interface without knowing how service requests are processed in the underlying layers. IoT users can access a service (e.g., reading or setting temperature conditions remotely or tracking and managing vehicles) using many platforms (e.g., laptops, smartphones, and smartwatches) via web portals or applications. Services vary based on the IoT scenario but can be categorized into four main classes as described in the following sections (Gigli and Koo, 2011).

3.3.4.1 Identity-Related Services

Identity-related services require an identifier that is embedded in a node and a reader device, such as an RFID device. Identity-related services can be active or

54 https://www.kaaproject.org/
55 http://devicehive.com/

passive (as discussed in the RFID subsections). These services are extremely important in IoT applications because they keep track of devices in large deployments (e.g., recording device maintenance and maintaining inventories of used parts and stock levels.) A package-tracking application is an example of this type of service.

3.3.4.2 Information Aggregation Services

Aggregation services summarize the collected raw sensory measurements from various types of sensors and networks that must be processed and reported to the IoT application. Load distribution among smart grids is an example of this type of service.

3.3.4.3 Collaborative Aware Services

Collaborative aware services are built on top of information aggregation services and are used to make decisions about the acquired data. This type of service can be found in smart homes, smart agriculture, and smart manufacturing, among other applications. For example, in a smart home, a security system and smart thermostats are employed together to improve security and energy efficiency based on sharing data between them (Anonymous, 2016f).

3.3.4.4 Ubiquitous Services

These are the most advantageous services of the IoT because they advance collaborative aware services to the next level by offering complete access to everything at (almost) any time and from anywhere. Access and control can be achieved using a computer, smartphone, or any smart device. Smart cities are examples of these services.

3.3.5 Business Layer Technologies

In this layer, unlike the application layer, service data and IoT environmental data, such as business models, flowcharts, and graphs, can be accessed. This access helps administrators in the design, analysis, implementation, evaluation, monitoring, and development of IoT systems because the output of each of the previously mentioned layers is analyzed in this layer to improve services and protect user privacy (Al-Fuqaha et al., 2015). The technologies in this layer can be divided into two categories based on their functions (i.e., semantics and Big Data analytics), as described in the subsequent sections.

3.3.5.1 Semantics

After data are captured by sensors, they can be analyzed to extract knowledge. Knowledge is crucial for enhancing services and obtaining useful results that cannot be discovered without examining all IoT environment data. In order to analyze data efficiently, it has to be well-formed using some semantics

technologies. Many semantic XML-based technologies can be used for knowledge extraction, resource discovery and usage, and modeling. An Efficient XML Interchange (EXI) is often used to resolve the performance issue that arises from the use of XML (e.g., XML parsing) to make it suitable for IoT applications (Al-Fuqaha et al., 2015). Many nodes from different brands are likely to generate data that has different form in each type of nodes. Such problem requires more work to unify the data so that it can be processed efficiently. Therefore, dynamic semantics is an important factor in improving the interoperability of the IoT. IoT technologies for semantics include the following:

- *Sensor Model Language (SensorML)*[56]*:* A standard model based on an XML schema to describe sensors and measurement procedures. SensorML is useful in IoT systems for creating electronic description sheets for sensor modules and collecting metadata to be used to discover sensor systems and observe processes. It also enables sensor networks to be autonomous because of the self-describing features of SensorML-supported sensors.
- *Media Types for Sensor Markup Language (SenML)*[57]*:* It is a new, simple model for acquiring sensed data and to control actuators. It provides semantics for the data and allows for additional metadata with links and extensions. This simple model can be used in many IoT applications (Keränen and Jennings, 2017). For example, a sensor, such as a humidity sensor, could use this media type in CoAP to transport the sensors' measurements.
- *IoT Database (IOTDB)*[58]*:* A new technology with unlimited expandability that supports semantics for providing formal definitions of all necessary items. Unlike the aforementioned technologies, IOTDB uses JSON dictionaries to manipulate and monitor nodes, which makes it relatively fast since JSON parsing is always more efficient than XML parsing. IOTDB is compatible with protocols, such as CoAP, and MQTT.
- *RESTful API Modeling Language (RAML)*[59]*:* This language is used to define HTTP-based APIs that represent most of the principles of Representational State Transfer (REST)[60]. Since it is RESTful, it is more likely to be used in IoT scenarios that are suitable for CoAP (as discussed in Section 3.3.3.2), where the network overhead is negligible.
- *Wolfram Data Drop*[61]*:* An open service that allows accumulating data of any type from anywhere (including IoT nodes) to prepare it semantically for instant computation, querying, analysis, visualization, or other operations.

56 http://www.opengeospatial.org/standards/sensorml
57 https://tools.ietf.org/html/draft-jennings-senml-10
58 https://iotdb.org
59 http://raml.org
60 Fielding, R.T. (2000) Representational State Transfer (REST). PhD dissertation. Available at https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm.
61 https://datadrop.wolframcloud.com

Computable data (i.e., collections and time series) are saved in named data bins in the Wolfram Cloud and are immediately accessible from all other systems/applications.

3.3.5.2 Big Data Analytics

IoT environments involve an immense number of sensors that collect vast amounts of data, resulting in extremely large datasets (i.e., Big Data) that may be computationally analyzed to extract knowledge, such as patterns, trends, and associations. The generated Big Data continues to increase as the IoT collects additional data. Therefore, special technologies that can address the ever-increasing amount of data are required. The most efficient strategy for addressing continuously increasing data is to process them in real time (streaming). Real-time processing enables up-to-date analytics that reflect recent changes in data and conserves storage. Machine learning can also utilize Big Data to make accurate predictions. Additionally, parallel computing can be employed using some IoT nodes (or fog computing units) to process data in parallel with back-end servers for better performance and load balancing. The following technologies aid in the generation of real-time (streaming) analytics from Big Data:

- *Apache Spark*[62]*:* A well-known open-source distributed processing technology that utilizes in-memory caching and features enhanced execution for better performance. Apache Spark supports streaming analytics, batch processing, graph databases, ad hoc queries, and machine learning. Although Spark has been employed in numerous IoT systems, it requires some extra effort to overcome some limitations, such as time delays in real-time processing (Freedman, 2016). Spark is a very flexible technology since it supports static and streaming data and many programming languages (as presented in Table 3.3).
- *Apache Apex*[63]*:* An open-source platform, similar to Spark, unifies stream and batch processing in a distributed, scalable, fault-tolerant, performant, stateful, and secure manner. Unlike Spark, it has a very low latency and processes big-data-in-motion (i.e., processes streaming data, including in-flight modification). Its ability for processing big-data-in-motion, enables it to be integrated easily with fog computing units. However, it lacks out-of-order processing and only supports Java as a programming language. Although it is a relatively new platform, it is widely adopted in many large-scale IoT projects, such as smart grids (Anonymous, 2016g; DataTorrent, 2016) and industrial applications (Apache Apex, 2016).
- *Apache Flink*[64]*:* An open-source platform that is conceptually very similar to Apex. However, it is older than Apex, it has not been used frequently in

62 http://spark.apache.org
63 https://apex.apache.org
64 https://flink.apache.org

Table 3.3 Comparison of Big Data streaming processing technologies.

Technology	API languages	QoS policies	Autoscaling	Latency	In-flight modifications
Spark	Scala, Java, R, Python	Exactly once At least once	Yes	Medium	No
Apex	Java	Exactly once At least once At most once	Yes	Very low	Yes
Flink	Scala, Java, Python	Exactly once	No	Low	No
Kafka	Java	At least once	Yes	Very low	Yes

real-life IoT projects due to its lack of autoscaling and limited message delivery guarantees (QoS).

- *Apache Kafka*[65]*:* A client library used to process and analyze the data stored in Kafka and either forward the output to an external system or write it back to Kafka. The first scenario allows Kafka to serve as a middle layer in which it collects the data generated by IoT sensors, then applies some operations to it (e.g., filtering and sorting), and finally streams it to other components in other frameworks (e.g., Flink and Spark).

Table 3.3 highlights the main differences between these technologies. The first column describes the languages that these Big Data frameworks support. This helps IoT adopters to choose Big Data frameworks based on their employee's expertise. The next column describes the delivery policies, which might be an important element in critical IoT scenarios in which "at least once" policy is appropriate. In contrast, in less critical scenarios (or for more efficiency), at most once would be sufficient. In this feature, Apex has an advantage over the others because it has the highest number of policies. Autoscaling is a very important feature for IoT because it ensures the efficiency of computation as extra resources are used only when needed. Latency can be relevant in some IoT scenarios (e.g., traffic control) to ensure the accuracy of real-time analytics. Finally, the last column indicates the ability of the framework to perform some computations on data before they reach the back-end servers, as in fog computing.

65 http://kafka.apache.org

3.4 IoT Platforms and Operating Systems

There are a vast number of IoT platforms and operating systems that can integrate many of the abovementioned technologies to provide IoT services. In the context of IoT, both terms (i.e., platforms and operating systems) are used interchangeably. However, there is a slight difference between them, that is, operating systems (i.e., embedded operating systems or RTOS) are focused only on some communication-related functions such as connecting sensors to the Internet and allowing data to be collected from sensors. Other functions, such as analytics, are often occurring, if needed, in partner systems. This makes operating systems suitable for small IoT applications. On the other hand, IoT platforms cover almost all functionalities from the five mentioned layers (e.g., communication, data exchange, analytics, etc.). Another advantage of IoT platforms is ease of setup and deployment. IoT platforms, in contrast to operating systems which require extra effort to integrate with other systems, enable IoT adopters to easily choose suitable technologies based on their needs and give them better compatibility and support. This section discusses only the popular generic IoT operating systems and platforms.

The following are two well-known C-based IoT operating systems:

- *RIOT*[66]*:* A lightweight operating system for memory-constrained, wireless networked systems with a focus on low-power IoT devices. RIOT supports a wide range of devices and offers higher complexity that allows for larger and time sensitive IoT applications.
- *Contiki*[67]*:* An open-source operating system for connecting memory-constrained, low-power wireless networked systems to the Internet. It is similar to RIOT but less advanced in terms of latency and performance (i.e., multithreading) capabilities. However, one of the main advantages of Contiki is its simulator (i.e., Cooja). Cooja helps researchers and developers to test their systems before purchasing devices. Since Contiki is relatively older than RIOT, it is widely used and hence, has an active community.

Other operating systems can also be used, such as Linux, but due to its requirement for larger ROM and RAM (i.e., ~1 MB), it is not suitable for tiny IoT devices. Table 3.4 compares main features in RIOT and Contiki operating systems. RIOT OS is more advanced because it supports multithreading as well as for its very low latency (almost real-time).Most IoT platforms are cloud based and provide IoT technologies for most functions mentioned in the previous section. The following are some platforms among the vast number of current platforms:

66 https://riot-os.org/
67 http://www.contiki-os.org/

Table 3.4 Comparison of IoT operating systems.

OS	Programming languages	Simulator	Required RAM, ROM	Multithreading	Real-time capabilities
RIOT	C, C++	None	1.5 KB, 5 KB	Yes	Yes
Contiki	Partial C	Cooja	2 KB, 30 KB	No	No

- *AWS IoT*[68]*:* A commercial IoT platform that exploits the popular AWS cloud services to provide everything that an enterprise needs (i.e., device management, visualization). Data ingestion, storage, processing, and visualization are handled by services such as Amazon Redshift, Amazon EMR, AWS Lambda, Amazon DynamoDB, Amazon Kinesis, and Amazon QuickSight.
- *IBM Watson*[69]*:* A commercial IoT platform that is strongly integrated with Bluemix to bring the power of cognitive computing and machine learning to IoT. This platform can be deployed on the cloud or on-premises in which onboarding of devices to the platform is automated using the SDKs and APIs. It also allows for blockchaining in which IoT integration with the evolving distributed ledger technology is based on HyperLedger.
- *ThingWorx*[70]*:* A commercial platform for the development of connected smart devices using an integrated IoT development tool that supports connectivity, production, analysis, and other areas of IoT. ThingWorx allows users to connect devices, establish a data source, establish device behaviors, and build an interface without any coding.
- *Bosch IoT Suite*[71]*:* A commercial flexible cloud-based IoT that allow developers to test the applications before implementing them, deploying them, and operating them under normal conditions. Its device management capabilities (i.e., executing software roll-out processes, connecting third-party systems and services, and analyzing data) can also be used stand-alone and on-premise.
- *Xively*[72]*:* A commercial cloud-based platform that makes the development of connected smart products for business easy due to its strong partnerships with hardware partners and one-click integrations with business tools. Xively enables visualization of data graphically in real time as well as updating devices remotely.
- *EVRYTHNG*[73]*:* A cloud-based platform to manage IoT identities for products in which all products are given a persistent, addressable web presence. It

68 https://aws.amazon.com/iot-platform/
69 https://www.ibm.com/internet-of-things/
70 https://www.thingworx.com/
71 https://www.bosch-si.com/iot-platform/bosch-iot-suite/homepage-bosch-iot-suite.html
72 https://www.xively.com/
73 https://evrythng.com/

Table 3.5 Comparison of IoT platforms.

Platform	Data exchange	Security	Integration	Device management
AWS IoT	MQTT, HTTP	TLS, SigV4[a], X.509[b]	REST API	Yes
IBM Watson	MQTT, HTTPS	TLS, IBM Cloud SSO[c], LDAP[d]	REST and Real-time APIs	Yes
ThingWorx	MQTT, AMQP, XMPP, CoAP, DDS, WebSockets[e]	ISO 27001[f], LDAP	REST API	Yes
Bosch IoT Suite	MQTT, CoAP, AMQP, STOMP[g]	Unknown	REST API	Yes
Xively	HTTP, HTTPS, WebSocket, MQTT	SSL/TSL	REST API	No
EVRYTHNG	MQTT, CoAP, WebSockets	SSL	REST API	No
Kaa	MQTT, HTTP	RSA and AES	REST API	Yes

a) http://docs.aws.amazon.com/general/latest/gr/signature-version-4.html
b) https://www.ietf.org/rfc/rfc5280.txt
c) https://www.ibm.com/security/cloud/cloud-identity-service/
d) https://tools.ietf.org/rfc/rfc4511.txt
e) https://tools.ietf.org/html/rfc6455
f) https://www.iso.org/isoiec-27001-information-security.html
g) https://stomp.github.io/

enables elastic semantic data store to customize these dynamic data profiles for any product so that authorized applications can interact with them and exchange data in real time during their lifecycle.

- *Kaa*[74]*:* An open-source cloud-based platform for managing IoT devices and analyzing generated data to provide complete end-to-end IoT solutions, connected applications, and smart products. It is compatible with virtually any connected devices and gateways. It enables devices to be used almost as plug and play units with minimal code.

Table 3.5 gives an overview of the main features of the popular IoT platforms. Data exchange protocols column indicates the protocols used to collect and send data among devices. The security column indicates security protocols used for authentication and confidentiality. The integration column shows the way

74 https://www.kaaproject.org/

that a platform can be integrated with other systems. Finally, device management indicates whether the platform enables devices to be remotely monitored, updated, disabled, and so on.

3.5 Conclusion

This chapter presented current and commonly employed technologies and protocols and their functionalities in a logical manner based on the IoT five-layer model. The perception layer technologies discussed are generic physical devices that can be used to send sensed data. The networking technologies mentioned in the second layer enable nodes to be identified and reached over a secure medium using special routing protocols, such as RPL. The technologies discussed in the middleware layer utilize physical components to provide services that are easily discoverable and organized for use by IoT users in the upper layers. The application layer provides the ultimate goal of IoT systems (i.e., services) to allow end users to access and use them on their favorite platform. Finally, the technologies in the business layer help system administrators to monitor and enhance IoT activities through analytics performed on collected data.

The technologies were discussed in a generic manner without suggesting any preferences because the most suitable IoT technologies cannot be recommended without knowing the requirements of the specific scenario in which they will be applied. For example, the IoMT probably requires low latency and secure technologies; hence, choosing technologies that have such features would be appropriate. The informative comparisons in this chapter highlighted the main differences among the various technologies.

Several challenges remain associated with the currently available IoT technologies. First, better horizontal integration among protocols is needed in each layer to enable existing IoT applications to collaborate and share resources. For example, in the middleware layer, a CoAP system should be able to integrate with DDS. This integration will enable collaboration among organizations and allow them to effectively share resources (e.g., sensors) or data. Second, service continuity for mobile Internet remains challenging because all current technologies rely on fixed points (e.g., cellular towers and border routers) for connectivity (Elsaleh et al., 2011). Third, IoT scalability and interoperability remain challenging as compatibility issues are inevitable in its current state because many different technologies are involved (Sarkar et al., 2014). Finally, security in the IoT is a distinct problem affecting currently available technologies due to the large-scale potential impact of successful attacks.

IoT enabling technologies are the key to building effective and successful IoT applications. As the number of technologies continues to expand, understanding the characteristics and limitations of currently available technologies is important because they provide the foundation for new technologies. Hopefully,

current challenges in the IoT will be minimized by the rapid advancement of IoT technologies driven by many organizations worldwide.

References

Ackerman, E. (2015) Disney seeks to make visible light communication practical. IEEE Spectrum: Technology, Engineering, and Science News, Available at http://spectrum.ieee.org/tech-talk/computing/networks/disney-seeks-to-make-visible-light-communication-practical (accessed Mar 30, 2017).

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., and Ayyash, M. (2015) Internet of Things: a survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys and Tutorials*, **17**(4), 2347–2376.

Al-Qaseemi, S.A., Almulhim, H.A., Almulhim, M.F., and Chaudhry, S.R. (2016) IoT architecture challenges and issues: lack of standardization. 2016 Future Technologies Conference (FTC), pp. 731–738.

Anonymous (2016a) GitHub – google/physical-web: The Physical Web: walk up and use anything. Available at https://github.com/google/physical-web (accessed Sep 23, 2016).

Anonymous. (2016b) Vulnerability Note VU#357851 - UPnP requests accepted over router WAN interfaces. Available at http://www.kb.cert.org/vuls/id/357851 (accessed Sep 24, 2016).

Anonymous. (2016c) OCF - UPnP+ Initiative. Open Connectivity Foundation (OCF).

Anonymous. (2016d) DDS-DLRL 1.4. Available at http://www.omg.org/spec/DDS-DLRL/1.4/ (accessed Sep 30, 2016).

Anonymous. (2016e) Raspberry Pi security system with motion detection/Camera. *Hackster.io*. Available at https://www.hackster.io/FutureSharks/raspberry-pi-security-system-with-motion-detection-camera-bed172 (accessed May 6, 2017).

Anonymous. (2016f) SimpliSafe | Home Security Systems. Available at http://simplisafe.com/simplisafe-now-works-with-nest (accessed Dec 2, 2016).

Anonymous. (2016g) IOT Big Data Ingestion and Processing in Hadoop by Silver Spring Network. Available at http://www.slideshare.net/ApacheApex/iot-big-data-ingestion-and-processing-in-hadoop-by-silver-spring-networks (accessed Oct 17, 2016).

Anonymous. (2017) UWB: Back from the dead, bound for IoT location-based services | FierceWireless. Available at http://www.fiercewireless.com/tech/uwb-back-from-dead-bound-for-iot-location-based-services (accessed May 31, 2017).

Apache Apex, (2016) GE IOT Predix Time Series & Data Ingestion Service using Apache Apex. Available athttps://www.slideshare.net/ApacheApex/ge-iot-predix-time-series-data-ingestion-service-using-apache-apex-hadoop (accessed Mar 31, 2017).

Bonomi, F., Milito, R., Zhu, J., and Addepalli, S. (2012) Fog computing and its role in the Internet of Things, in Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, New York, NY, USA, pp. 13–16.

DataTorrent, *Meetup + Slides: GE IOT Predix Time Series & Data Ingestion Service Using Apache Apex (Hadoop)*, 2016.

Duffy, J. (2014) SDN vital to IoT. Network World, Available at http://www.networkworld.com/article/2601926/sdn/sdn-vital-to-iot.html (accessed Nov 27, 2016).

Elsaleh, T., Gluhak, A., and Moessner, K. (2011) service continuity for subscribers of the mobile real world Internet. 2011 IEEE International Conference on Communications Workshops (ICC), pp. 1–5.

Fontana, R.J. (2004) Recent system applications of short-pulse ultra-wideband (UWB) technology. *IEEE Transactions on Microwave Theory and Techniques*, **52**(9), 2087–2104.

Gigli, M. and Koo, S. (2011) Internet of Things: services and applications categorization. *Advances in Internet of Things*, **1**(2), 27–31.

Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M. (2013) Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, **29**(7), 1645–1660.

Hassan, Q.F., Riad, A.M., and Hassan, A.E. (2012) Understanding cloud computing, in Yang, H. and Liu, X. (eds), *Software Reuse in the Emerging Cloud Computing Era*, IGI Global.

Jacquenet, C. and Boucadair, M. (2016) A Software: Defined Approach to IoT Networking. September.

Karagiannis, V., Chatzimisios, P., Vazquez-Gallego, F., and Alonso-Zarate, J. (2015) A survey on application layer protocols for the internet of things. *Transaction on IoT and Cloud Computing*, **3**(1), 11–17.

Keränen, A. and Jennings, C. (2017) SenML: simple building block for IoT semantic interoperability. Available at https://www.iab.org/wp-content/IAB-uploads/2016/03/IAB_IOTSI_Keranen_Jennings_SenML.pdf (accessed May 4, 2017).

Li, S., Xu, L.D., and Zhao, S. (2015) The Internet of Things: a survey. *Information Systems Frontiers*, **17**(2), 243–259.

Masinter, L., Berners-Lee, T., and Fielding, R.T. (2016) Uniform Resource Identifier (URI): Generic Syntax. Available at https://tools.ietf.org/html/rfc3986 (accessed Sep. 1, 2016).

Freedman, M. (2016) Spark streaming and IoT. Available at https://spark-summit.org/east-2016/events/spark-streaming-and-iot/ (accessed Oct 17, 2016).

Nokia (2017) LTE evolution for IoT connectivity white paper. Available at http://resources.alcatel-lucent.com/asset/200178 (accessed May 3, 2017).

Prasad, R. (2014) *5G: 2020 and Beyond*, River Publishers.

Raza, S., Chung, T., Duquennoy, S., Dogan, Y., Voigt, T., and Roedig, U. (2011) Securing Internet of Things with Lightweight IPsec.

Saint-Andre, P. and Hildebrand, J. (2011) Message Delivery Receipts. Available at https://xmpp.org/extensions/xep-0184.html (accessed Mar 30, 2017).

Sarkar, C., Nambi, S.N.A.U., Prasad, R.V., and Rahim, A. (2014) A scalable distributed architecture towards unifying IoT applications. 2014 IEEE World Forum on Internet of Things (WF-IoT), pp. 508–513.

Stephen C. (2016) 10 things to know about the October 21 IoT DDoS attacks. *We Live Security*, (posted on October 24, 2016 at 07:16 pm). Available at http://www.welivesecurity.com/2016/10/24/10-things-know-october-21-iot-ddos-attacks/ (accessed Dec 01, 2016)

StormMQ, (2017) A Comparison of AMQP and MQTT white paper. Available at https://lists.oasis-open.org/archives/amqp/201202/msg00086/StormMQ_WhitePaper_-_A_Comparison_of_AMQP_and_MQTT.pdf (accessed May 4, 2017).

Waher, P. and DOI, Y. (2015) Efficient XML Interchange (EXI) Format. Available at http://xmpp.org/extensions/xep-0322.html (accessed Sep 27, 2016).

West, D.M. (2016) How 5G technology enables the health internet of things. Center for Technology Innovation Brookings, vol. 3.

Weyrich, M. and Ebert, C. (2016) Reference architectures for the Internet of Things. *Software IEEE*, **33**(1), 112–116.