Chapter 1
A Review of IoT Technologies, Standards, Tools, Frameworks and Platforms

Eldar Sultanow and Alina Chircu

Abstract In this contribution, we present an integrated view of the technologies, standards, tools, frameworks and platforms that support the end-to-end Internet of Things (IoT) solutions in general terms and highlight specific Industrial IoT (IIoT) solution components. Our study goes beyond existing research, including our own previous work, by focusing on all relevant IoT/IIoT solution components relating to development and operation. Specifically, we discuss the communication standards, messaging protocol standards, and communication platforms; device control, integration and simulation frameworks; tools and frameworks for modeling, development and deployment; and IoT cloud integration platforms that support IoT solutions. By highlighting the features as well as the advantages and limitations of different IoT solutions, this technical analysis can prove useful to IoT practitioners designing IoT and IIoT systems with diverse requirements; to students further learning about IoT/IIOT vision; and to researchers interested in understanding the current limitations of the IoT/IIoT landscape and developing new standards, tools, frameworks and platforms for future application.

Keywords Internet of things · IoT · Industrial IoT · IIoT · Reference architecture · IoT development · Standards · RAMI 4.0 · Industry 4.0

1.1 Introduction

The Internet of Things (IoT), a term coined in the 1990s in the context of supply chains [1], is one of the most exciting technology developments today. It promises to embed sensors into physical things (personal devices, industrial machines, vehicles, appliances, and the like) and enable them to record, process, and communicate their status data (position, movement, temperature, operating status, errors, etc.) with other

E. Sultanow
Capgemini Germany, Bahnhofstraße, 11C, 90402 Nuremberg, Germany
e-mail: eldar.sultanow@capgemini.com

A. Chircu (✉)
Bentley University, 175 Forest Street, Waltham, MA 02452-4705, USA
e-mail: achircu@bentley.edu

things or with Internet servers (directly or through gateways) for further processing. The IoT market size is predicted to reach many hundreds of billions of dollars by 2020, with the USA, China, Germany, and UK leading the growth [2].

The Industrial IoT (IIoT), also known as the Industrial Internet, the Internet of Industrial Things, and Industry 4.0, focuses on the IoT technologies that connect industrial machines among themselves and to the Internet. This enables automated instrumentation, reporting, and even manufacturing [3, 4]. With 2.6 billion connections between industrial machines predicted by 2020, IIoT is one of the top growth sectors in the IoT scenario [2, 5], leading to an expected market size of over $100 billion by 2020 [5]. The top industries anticipated to benefit from the IIoT vision are discrete manufacturing, transportation, logistics and supply chains, utilities such as electricity, and health care [3–9]. IIoT applications related to smart factories, smart warehousing, predictive and remote maintenance, freight, goods and transportation monitoring, smart utility metering, smart grids, smart cities, smart farming, livestock monitoring, asset tracking and performance management, industrial environment monitoring, safety and health monitoring, and many others, are predicted to become more efficient and productive and, in turn, enable digital transformation in companies, in fact the entire industries [3, 10, 11].

While IoT and IIoT share the same basic building blocks including technologies, the need for integrated solutions is much more acute in the IIoT space, where standards, interoperability, and integration with legacy technologies are some of the most important factors for successful technology development, implementation, management, and adoption [3, 5, 8, 12, 13]. Not surprisingly, leading global companies such as Bosch, Dell, GE, Huawei, IBM, Siemens, Mitsubishi, Cisco, China Mobile, Boeing, Intel, SAP, Ericsson, and many others are taking an active interest in defining standards, best practices, and processes through IIoT consortia and other collaboration initiatives (as described in the next section).

In this chapter, we present an integrated view of the technologies, standards, tools, frameworks and platforms that support end-to-end IoT (and the subset of IIoT) solutions. We build on our ongoing research focusing on specific industries (pharma, health care, and life sciences) and generic architectural frameworks [14, 15]. In particular, we significantly extend our recent work on an integrated IoT reference architecture [16]. The chapter's novel contribution is a detailed discussion of the components necessary for a complete IoT solution and how they fit into the IoT development, implementation, and management ecosystem. The need for such an approach has been advocated by many other IoT researchers [5, 8, 12, 13]. While simpler analyses of individual IoT components have been published (see, for example, [13]), to our knowledge, no other authors have discussed all the different technologies, standards, tools, frameworks and platforms that support an end-to-end IoT solution in the same paper. Thus, this book chapter can serve as a reference for practitioners developing new IoT solutions. It can also help researchers identify unmet needs and design improved or novel IoT technologies, standards, tools, frameworks and platforms.

This chapter covers several important topics including: architectures and frameworks; communication protocols and device connectivity; and sensors and actuators, among others. The chapter is organized as follows: Sect. 1.2 examines the related work, Sect. 1.3 presents the analysis of IoT technologies, standards, tools, frameworks and platforms, and Sect. 1.4 presents the conclusions and suggestions for future research.

1.2 Related Work

Industrial Internet of Things (IIoT) is the application of IoT technologies to industrial settings such as manufacturing. Based on a review of existing research, IIoT can be defined as *a system comprising networked smart objects*, *cyber-physical assets*, *associated information technologies*, *and optional cloud or edge computing platforms*, *which enable real-time*, *intelligent and autonomous access, collection*, *analysis*, *communication*, *and exchange of process*, *product and/or service information*, *within the industrial environment*, *so as to optimize overall production value*. *This value may include*: *improving product or service delivery*, *boosting productivity*, *reducing labor costs, reducing energy consumption*, *and reducing the build-to-order cycle* [17]. IIoT is closely related to the Industrie 4.0 (or Industry 4.0) concept promoted by the German government as a way of developing the German economy through a fourth industrial revolution (Industry 4.0) [17–19]. The Industry 4.0 definition emerging from published research is *an integrated adapted*, *optimized*, *service-oriented*, *and interoperable manufacturing process which is correlated with algorithms*, *big data*, *and high technologies* [20]. Other related concepts include cyber-physical systems (which combine physical objects and processes with real-time data collection and autonomous digital monitoring and control capabilities), and smart factories or cyber-physical production systems (which are autonomously controlled flexible manufacturing facilities) [17–20].

Over the last few years, researchers and practitioners alike have become more interested in developing reference architectures for IoT and IIoT [18, 19]. Such frameworks describe the necessary components for building integrated end-to-end IoT solutions in general, or building industry-specific IIoT applications.

Generic architectures usually organize the components necessary for building IoT solutions in several distinct layers. Lin et al. [21] present a four-layer architecture (including perception, network, service, and application layers), describe key technologies and standards in each layer, analyze security and privacy issues, and propose a way of processing the IoT data using fog/edge computing. Similarly, Xu et al. [22] discuss a four-layer architecture (including sensing, networking, service, and interface layers) as well as key enabling technologies and key applications for industries such as health care, food supply chains, and transportation and logistics, among others. Specific architectures are also being developed in many areas, including those relevant for IIoT such as health care, manufacturing, transportation, and logistics.

In the healthcare domain, Pang et al. [23] propose an architecture for in-home healthcare devices that includes sensors, network connections to an in-home healthcare system, and cloud connections to other information systems such as hospital systems. Ramirez et al. [24] describe an IoT architecture for sensory impaired individuals, where sensors, actuators, and monitoring devices such as mobile phones and tablets can transmit data over wired or wireless networks to dedicated applications that interpret the data and intelligently respond to user needs. Dhariwal and Mehta [25] propose a 3-layer architecture for a smart hospital that includes a perception layer (which collects data from patients, doctors, and nurses and transmits this data to network), a network layer (which transmits real-time data within the network and integrates various sources of data), and an application layer (which manages hospital operations, equipment, finances and provides decision support based on analysis of data about diseases, patients, clinics, department, medicines, etc.).

In the manufacturing domain, Zhang et al. [26] propose a generic architecture to attach sensing capabilities to various resources in a manufacturing environment, capture data in real time during the production process, filter, and process data into meaningful metrics, and provide real-time analysis and feedback based on the data. Zancul et al. [27] develop an architectural model relevant for product–service systems and identify key IoT impacts in processes such as remote machine setup, maintenance (both corrective and predictive), supply chains, product pricing, and information reporting.

Other researchers focus on IoT architectures from a specific functionality perspective, such as energy-efficient IIoT [28] or secure IoT [29, 30]. Last, but not least, there are also efforts to define parts of IIoT architectures, such as analytical frameworks for describing the IIoT devices [17].

On the practitioner side, IIoT communities of interest and reference models are also proliferating. For example, the Industrial Internet Consortium (IIC) has defined the Industrial Internet Reference Architecture (IIRA), a standard-based open architecture for IIoT systems applicable across many industries (e.g., energy, health care, manufacturing, transportation, etc.) [31].

IIRA includes four viewpoints: business (focused on identifying the business stakeholders and their vision, values and objectives), usage (focused on the expected system use), functional (focused on the functional components of the system and their interconnections), and implementation (focused on technologies that support the functional components).

Industrial Internet Consortium (IIC) has developed an Industrial Internet Connectivity Framework (IICF), which organizes connectivity technologies into a stack model and proposes achieving interoperability among various standards through gateways [31].

In the Industry 4.0 area, Platform Industrie 4.0 has developed the Reference Architecture Model for Industrie 4.0 (RAMI 4.0) which is a detailed service-oriented architecture for manufacturing. It focuses on several axes: factory (which defines how products and manufacturing systems interact in an enterprise), product life cycle (which captures the development, production, maintenance, usage and end of life stages of a product), and architecture (based on six layers describing both the real and digital world) [31].

While IIRA and RAMI 4.0 each have a different focus (broad industry applicability versus manufacturing only), efforts to compare and map their components to each other are already underway in order to ensure interoperability between different IIoT systems built on these two different architectures [31].

Major technology providers, such as IBM, are also active in the IoT architecture area, proposing both general and specific reference architectures for IoT and IIoT [32]. IBM's IoT reference architecture consists of five layers (user, proximity network, public network, provider cloud, and enterprise network) and has applicability to multiple industries. IBM's Industrie 4.0 reference architecture is specific to manufacturing and consists of three manufacturing-specific layers: edge, plant, and enterprise [32].

In addition, several other organizations are involved in developing IIoT standards and tools that can be used to build IIoT systems. The Object Management Group (OMG) has developed the Data Distribution Service (DDS) standard, which is promoted as *the first open international middleware standard directly addressing publish/subscribe communications for real-time and embedded systems*, which makes it ideal for *high-performance*, *highly scalable Industrial Internet of Things (IoT) and large-scale Consumer IoT application environments which require real-time data communication exchange* [33]. The Open Connectivity Foundation is focused on IoT device interoperability for consumers, businesses and industries and on *delivering a standard communications platform*, *a bridging specification*, *an open-source implementation and a certification program allowing devices to communicate regardless of form factor*, *operating system, service provider*, *transport technology or ecosystem* [34]. The Internet of Things Consortium (IoTC) is focused on IoT ecosystem *interoperability and usability*, *data openness and security*, *and market development* and provides information on IoT use cases, IoT hardware and software, as well as IoT vendors [35]. Last, but not least, the Eclipse Foundation, a nonprofit organization where individual developers and companies from many industries collaborate on open-source projects (providing runtimes, tools, and frameworks for many domains), has emerged as a major IoT player over the last few years. Its Eclipse IoT Working Group involves companies such as Bosch, Huawei, IBM, Intel, Nokia, SAP, and Siemens and has generated many relevant open-source projects for the IoT landscape [36].

1.3 Analysis

Our review of current research reveals that the existing architectures provide only a partial overview of the components necessary for developing an end-to-end IoT solution. It is usually just for sensing and networking technologies (in generic models) and just for industry-specific functionality (in specific models). To address this gap, we developed an integrated IoT reference architecture [16] focusing on the entire IoT life cycle, from IoT application development to operations (solution in action). Our proposed architecture senses the environment and collects data through IoT-enabled smart devices, transmits it through communication standards, protocols and platforms, processes it using back-end systems, and uses it in front-end applications. Figures 1.1, 1.2, and 1.3 illustrate a detailed graphical representation of the reference architecture, organized in three components:

- devices and associated communication standards—refer to Fig. 1.1
- protocols, back end, and front end as presented in Fig. 1.2
- development platforms as shown in Fig. 1.3.

Taken together, Figs. 1.1 and 1.2 depict the necessary elements for an IoT end-to-end solution, which can be visualized as a stack consisting of a bottom layer describing the devices (hardware, firmware, and runtime environment), a layer of communication standards (as shown in Fig. 1.1), and layers of messaging protocol standards, back-end and front-end tools, frameworks and platforms (as shown in Fig. 1.2). Figure 1.3 depicts development platforms that address the description and integration of the devices shown in Fig. 1.1, as well as the setup of the back-end platforms and the development of IoT applications shown in Fig. 1.2. Interested readers should refer to our existing research [16] for an explanation of how the reference architecture was developed.

In this chapter, we extend our previous work [16] by analyzing the various IoT technologies, standards, tools, frameworks and platforms included in our architecture. These include connected devices, communication standards, messaging protocol standards, communication platforms, device control, integration and simulation frameworks, tools and frameworks for modeling, development and deployment, and

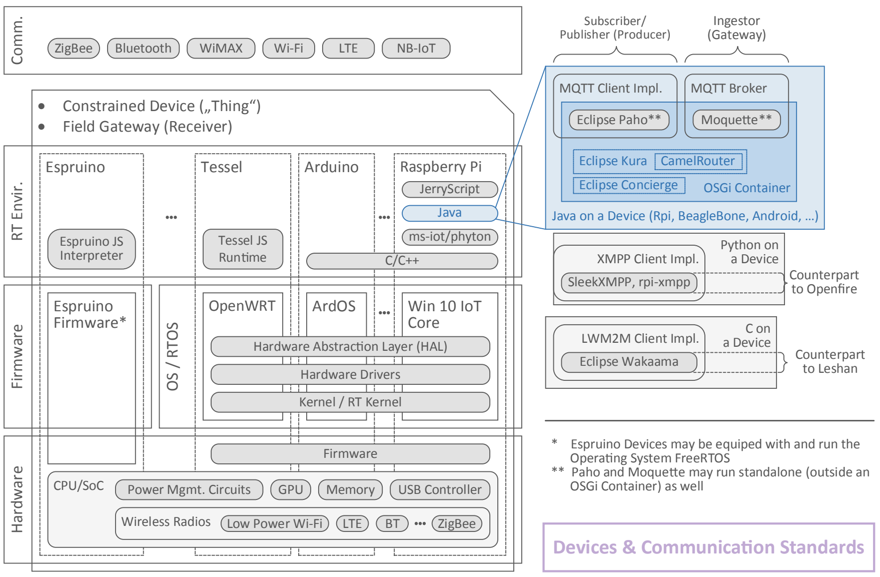

Fig. 1.1 Reference architecture for IoT—devices and communication standards. Adapted from [16]

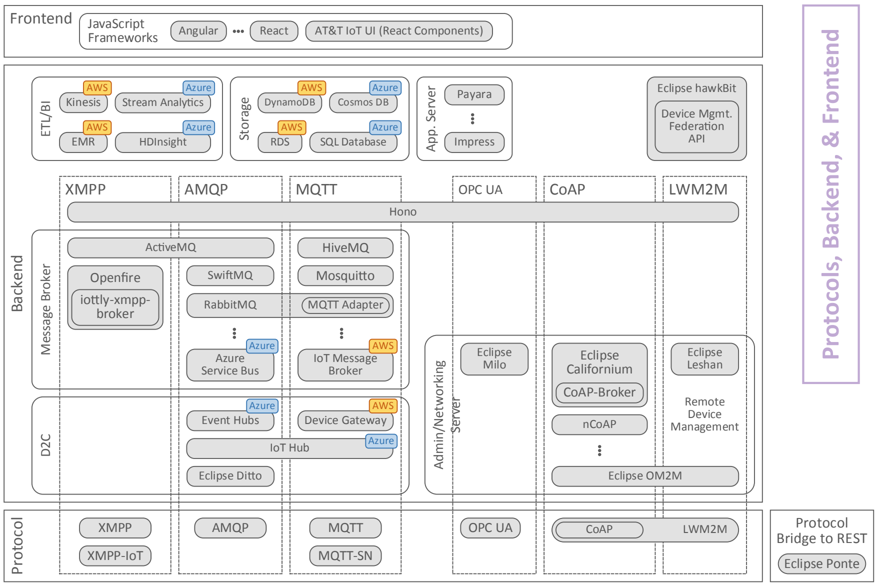

Fig. 1.2 Reference architecture for IoT—protocols, back end, and front end. Adapted from [16]

IoT cloud integration platforms. When appropriate, we also highlight specific IIoT solution components. We focus our analysis on characteristics relevant for developers and researchers interested in choosing one option over another (for example, bandwidth, energy consumption, security for standards, or specific technical features for platforms and frameworks), and present advantages and disadvantages as applicable.

1.3.1 IoT Devices

IoT devices, or *things* in the IoT, are the first building block of the IoT infrastructure. Any physical object (including human users of the IoT) can become IoT-enabled with the addition of IoT hardware that provides functionality ranging from simple sensing abilities to more sophisticated processing and networking capabilities. IoT hardware includes both embedded systems and boards (such as Arduino, Raspberry Pi, Tessel, Espruino, Pinoccio, Beaglebone Black, and others) and stand-alone devices (such as Samsung Gear or FitBit in the consumer-things space) [37]. Embedded systems consist of a microcontroller (with one or more processors, memory, graphics processing unit, general-purpose input/output interfaces, and specific interfaces such as wireless networking, camera, or USB) as well as other parts (such as a power source, sensors for light, heat, motion, sound or other environmental inputs, analog-to-digital and digital-to-analog converters, various actuators, motors, smart materials,

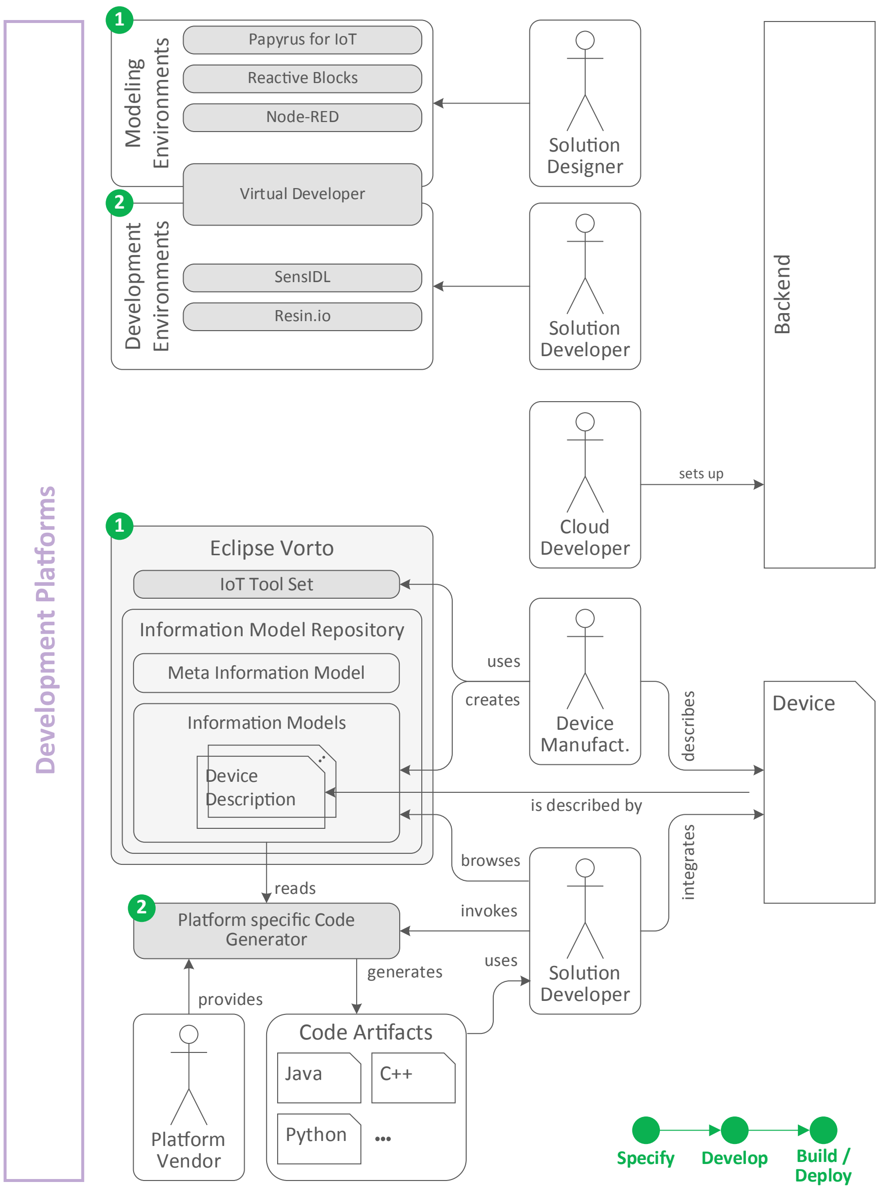

Fig. 1.3 Reference architecture for IoT—development platforms. Adapted from [16]

and devices that can control the environment usually by converting a source of energy into motion).

In this context, many different systems exist; and IoT designers can choose the best platform for a specific application based on a variety of factors. Interested readers can consult the most recent detailed comparisons provided by Singh and Kapoor [37], which is based on data collected in 2017.

1.3.2 Communication Standards

Communication standards are essential for enabling IoT devices to reliably communicate with each other and with other entities on the Internet. A comparison of the most widely used communication standards based on current practitioner analyses [38–45] is presented in Table 1.1.

One of the most widespread communication standards is Wi-Fi (or WiFi), which is a wireless local area network (WLAN) standard defined by IEEE 802.11 specification. This standard enables connections directly with a server, such as in the case of Sonoff WiFi Power Switches, which can send their readings of energy consumption directly to the OpenWrt Router [38]. OpenWrt is a Linux distribution for embedded systems like WLAN Router (e.g., Linksys provides such a router with WRT3200ACM which is enabled with OpenWrt Open Source).

Another standard is ZigBee, an IEEE-802.15.4 specification for wireless personal area networks well-suited for low-power devices and for building automation, sensor networks, and light technology applications. Although ZigBee's focus is on short-range networks, bigger ranges up to several kilometers are still possible.

Bluetooth is another standard (based on IEEE 802.15.1 specifications) for data transfer between devices over short distances. Typical deployment areas are mobile loudspeakers, connections between a smartphone and PC or between a smartphone and an onboard computer in a car, or wireless mouse and keyboard connections to a computer.

There is also the WiMAX/IEEE-802.16 standard, which is used for example by Sigfox and LoRa [39] and popular mobile standards like LTE—the fourth generation (4G) high-speed mobile communication standard. For a long time, the mobile phone industry did not show any interest in industrial wireless IoT applications, but the 3GPP (3rd Generation Partnership Project—the worldwide cooperation of standardization bodies for mobile industry) has, for the first time, standardized variations such as NB-IoT specifically for machine-to-machine (M2M) wireless communication. NB-IoT, which stands for Narrowband IoT, is a new wireless standard used, among others, for smart gas and water meters and for smart city applications (management of street lights, intelligent parking guidance systems, etc.) [40]. NB-IoT is similar to LTE-M, another cellular IoT standard designed for low-power, low-range applications [41].

Other emerging communication standards include LoRa, Sigfox, WAVIoT, DASH7, LTE-M (LTE M2M), NB-LTE-M, Weightless, 6LoWPAN (IPv6 over LWPAN), Z-Wave, Symphony, and Wavenis [42].

Table 1.1 Comparison of IoT communication standards (based on [38–45])

Standard	Frequency range	Data rate	Bandwidth	Coverage	Energy need	Advantages	Disadvantages
Wi-Fi	2.4–5.9 GHz Bands	Variable; 1.5–54 Mbps (IEEE 802.11a-1999 specification), but higher data rates up to 7 Gbps possible (see for example the IEEE 802.11ad specification)	Variable, up to 450–600 Mbps for 2.4 GHz Wi-Fi and up to 1300 Mbps for 5 GHz Wi-Fi	50–100 m	High energy need	Very simple setup, most robust security available to date, transparent implementations, multiple clients can connect to an access point simultaneously (depending on the max allowed for device), very high data rates, de facto standard for wireless Internet connectivity worldwide	Hotspots require a lot of power (i.e., constant energy source), adapters and drivers require memory and processing resources unavailable in simple appliances, and signals cannot go through walls without losing strength thus limiting range to within a building

(continued)

Table 1.1 (continued)

Standard	Frequency range	Data rate	Bandwidth	Coverage	Energy need	Advantages	Disadvantages
ZigBee	2.4 GHz	250 Kbps	2 MHz	50 m, several km possible	Low energy need	Low-power consumption; smart time allocation for fixed bandwidth communications, low cost, low latency (< 30 ms for device discovery, 15 ms for wake-up and 15 ms for active channel access) can be used to create large networks with up to 254 devices connected to a master, implements AES-128 for data encryption and security	Short range and low data transfer speed

(continued)

Table 1.1 (continued)

Standard	Frequency range	Data rate	Bandwidth	Coverage	Energy need	Advantages	Disadvantages
Bluetooth	Variable (2402–2480 MHz or 2400–2483.5 MHz) including guard bands 2 MHz wide (bottom) and 3.5 MHz wide (top)	Depends on standards: BT 3 & 4 up to 24 Mbps; BT 2 up to 3 Mbps; BT 1 up to 700 Kbps	Up to 1 Mbps (BT 4)	1–100 m (typical 10 m)	Medium to very low; BT 4 Bluetooth Low Energy (BLE) has very low energy need	Widely used standard makes it convenient to connect various devices, cost-effectiveness makes implementation in low-end devices possible, very easy and convenient to use, BLE paves the way for more possibilities as its low energy needs make it suitable for more low-power devices	Security concerns (devices can be hacked into easily), only one ongoing connection at a time, limited in range (ideal only for communication within a room, more range requires a lot more power), BLE bandwidth capability limited and transfer rate lower

(continued)

Table 1.1 (continued)

Standard	Frequency range	Data rate	Bandwidth	Coverage	Energy need	Advantages	Disadvantages
WiMAX	2–11 GHz	124 Mbps	Adjustable 1.25 M to 20 MHz	30–50 km	High energy need	Long range; symmetrical bandwidth over long range; hundreds of users can be served from a single WiMAX station; operates on an unlicensed spectrum of frequency	Serving lots of clients results in poor bandwidth; can be affected by weather conditions, high latency, line-of-sight required for longer range, signal prone to distortion due to noise

(continued)

Table 1.1 (continued)

Standard	Frequency range	Data rate	Bandwidth	Coverage	Energy need	Advantages	Disadvantages
LTE	Variable; in Germany, the LTE frequencies in the 800 MHz, 1800 and 2600 MHz ranges are used for the 4G mobile radio	Variable; usually 100 Mbps (download), 50 Mbps (upload)	Variable; 100 Mbps	Variable; 2–10 km; 30 or 75 km possible (Australia)	Low energy need	High speed (due to increased bandwidth) is advantageous for mobile devices; more coverage than other systems such as WiFi, which forces users to depend upon hotspots in each area they visit; 4G networks offer complete privacy, security, and safety	Connectivity is still limited to certain specified carriers and regions; new equipment would have to be installed in order to supply LTE services—however, the hardware compatible with 4G networks is available at much cheaper rates today than in the past

(continued)

Table 1.1 (continued)

Standard	Frequency range	Data rate	Bandwidth	Coverage	Energy need	Advantages	Disadvantages
NB-IoT	LTE in-band, guard band, stand-alone; in Europe, Telekom will use 800 and 900 MHz frequencies	250 Kbps	180 kHz	Gains of 20 dB over LTE; covers distance of less than 22 km	Low energy need (10-year battery life)	Low-power consumption (expected) moderate module costs, physical infrastructure already in place, ensuring fast rollout and market readiness, telco provided, proved ability to provide mass mobile networks with high service level	Less coverage than leading standards (Europe), less roaming support (2G sunset and available LTE infrastructure may boost licensed standards), early adoption/not yet mature; pricing models evolving; hard to build business case; no voice transmission support

1.3.3 Messaging Protocol Standards

Messaging protocols define the syntax and semantics of the data being transferred between IoT devices and servers. They focus on delivering entire messages, rather than on just transferring data without worrying about its meaning, as communication standards often do. Table 1.2 provides a comparison of most widely used protocol standards based on current practitioner analyses [46–57]. A brief description of these protocols is also provided below.

MQTT (Message Queuing Telemetry Transport) is an IoT messaging protocol introduced in 2011 that is maintained and enhanced by the independent Organization for the Advancement of Structured Information Standards (OASIS). It is highly efficient and scalable (up to several hundred thousand clients per server) and has a very small protocol overhead. MQTT implements a publish/subscribe messaging model on top of the widely known TCP/IP standards. It is specifically designed for machine-to-machine (M2M) use cases and provides reliable transfer of messages over unreliable (such as mobile or limited bandwidth) networks and for the high-performance deployment of IoT devices with low memory and lower processing power [46]. It is useful in the automobile industry (for connected cars), in the energy sector as well as in the field of building automation. Since 2016, MQTT is also an ISO standard (ISO/IEC 20922). The requirement to use an existing network such as TCP/IP limits the application of MQTT to devices that can support such a network. To address this limitation, MQTT-SN (MQTT for Sensor Networks) was developed as a version of MQTT designed specifically for wireless sensor networks (consisting of very simple, low-cost devices with limited power, processing, and storage capabilities) and their characteristics (high failure rates, lower transmission rates, shorter message lengths) [47]. Originally implemented on top of the ZigBee open communication standard, MQTT-SN is in fact designed to work on any underlying network. It is very close to MQTT but has several differences that enable it to support shorter message lengths and lower bandwidths [47].

AMQP (Advanced Message Queuing Protocol) is an open standard for a communication protocol between a client and a message broker, or between various message brokers. AMQP-enabled brokers include, for example, RabbitMQ, Apache ActiveMQ, Apache Qpid, SwiftMQ, Microsoft Azure Service Bus, and Red Hat Enterprise MRG. AMQP was created by a consortium of software companies and financial institutions, which include Microsoft, Red Hat, Cisco, Bank of America, Goldman Sachs, Credit Suisse, and many others. Microsoft has integrated the AMQP protocol directly into its Azure cloud solution.

XMPP (eXtensible Messaging and Presence Protocol), formerly known as Jabber, is an XML-based communication protocol frequently used by instant messengers for chat applications. It is less used in the field of IoT due to large protocol overheads caused by XML [46] and has even been phased out of chat applications (such as in the case of Facebook). XMPP has enhancements designed especially for IoT use cases. These, however, are supported by very few servers and clients. The GitHub-project XMPP-IoT provides a collection of such enhancements, including sensor discovery

Table 1.2 Comparison of IoT messaging protocol standards (based on [46–57])

Standard	Architecture	Underlying transport layer	Communication	Overhead	Security	Advantages	Disadvantages
MQTT	Asynchronous message exchange	TCP	Unicast	Small	Optional TLS, simple user-name/password authentication, SSL for data encryption	Minimal protocol overhead, several hundreds of thousands of clients per server, features designed for IoT use cases, suitable for resource-limited devices, client implementations available for all common languages	Pure request/response architectures can only be implemented in MQTT with additional effort, since it uses TCP connections to a MQTT broker, an always-on connection will limit the time the devices can be put to sleep
MQTT-SN	Asynchronous message exchange	Any	Unicast, multicast	Small	Optional TLS	Many-to-many communication protocol, credibility—supported by IBM, Eurotech, Cisco and Red Hat, open nature, open source	Lacks support for labeling messages, making it difficult to understand, must be familiar with the message formats to enable communication

(continued)

Table 1.2 (continued)

Standard	Architecture	Underlying transport layer	Communication	Overhead	Security	Advantages	Disadvantages
AMQP	Asynchronous message exchange	TCP	Unicast, multicast	Large	SASL authentication, TLS for data encryption	Many possibilities for message delivery (beyond Pub/Sub) suitable for broad range of messaging-middleware infrastructures and peer-to-peer data transfer	Overhead (smallest packet size 60 bytes) affects the expended effort in a large number of devices and messages, not easy to implement
XMPP	Synchronous message exchange	TCP	Unicast, multicast	Large	TLS and SASL	Clients available in many programming languages, many features, many optimized for instant messenger use cases	High protocol overhead, high fragmentation of server and client features, no end-to-end encryption, no QoS functionality

(continued)

Table 1.2 (continued)

Standard	Architecture	Underlying transport layer	Communication	Overhead	Security	Advantages	Disadvantages
CoAP	REST; layered approach	UDP	Unicast, multicast	Medium	Optional DTLS	Very efficient, URI and content-type support, open standard, designed for web interoperability, easily translates to HTTP, easy migration from HTTP, can be used on top of a variety of different packet-based communication protocols	Challenges of Network Address Translation (NAT) through UDP, few client and server implementations, 1-to-1 protocol—broadcast capabilities not native, best suited for devices that support UDP or a UDP analog
HTTP	REST; layered approach	TCP	Unicast	Large	Typically based on SSL or TLS	Libraries for almost all programming languages available, easy to learn.	High protocol overhead, no server push communication natively possible (only by means of long polling, server sent events or WebSockets), no possibility to map 1/N communication

(continued)

Table 1.2 (continued)

Standard	Architecture	Underlying transport layer	Communication	Overhead	Security	Advantages	Disadvantages
OPC UA	Service-oriented architecture (SOA); layered approach	TCP, UDP also possible	Unicast, multicast	Medium	Integral part of specification and communication stacks, OAuth 2.0, exchange of certificates for identification of application instances, SSL/TLS, WS Secure Conversation (UA XML), UA Secure Conversation (UA Binary)	Many features, open source, addresses security comprehensively, defines communications from the application to the transport layer making it very interoperable between vendors, interoperability to the device and enterprise levels	Not easy to implement, primarily beneficial for tying PLC and sensor data into existing industrial applications like SCADA and MES systems, where OPC and OPC UA connectivity are already available
LWM2 M	REST; layered approach	CoAP over UDP, TCP and SMS, LWM2M over MQTT possible	Unicast, multicast	Small	Secure communications between client and server using DTLS	Very flexible in utilization of underlying transport layer, widely used and established standard for remote device management, supports both powerful embedded devices and low-power ones	Implementation dependent on CoAP, needs CoAP expertise, data model is flat and simple (no complex trees) (this might be also considered as an advantage)

in sensor networks, efficient multicasting of sensor data, efficient data compression, interacting with humans and systems via chat messages, and many others [54].

CoAP (Constrained Application Protocol) is often described as the *HTTP for IoT*. It implements the request/response pattern—the most basic and widespread of the client–service interaction patterns. CoAP is more efficient than HTTP as the protocol is binary including all the headers and is thus suitable for energy-constrained wireless sensor networks. However, it lacks security features, requiring additional protocols for securing the network [55].

HTTP (Hypertext Transfer Protocol), the bedrock of all protocols, is still the first choice for request/response-based IoT communication [46]. Due to very high protocol overheads (since HTTP is completely text-based), it is rather unsuitable for use cases where the volume of data transferred matters (for example if mobile/bandwidth-constrained networks are involved). Furthermore, the request/response model allows 1-to-1, but not 1-to-many communication. Therefore, HTTP is also not suitable when a message has to be sent to several recipients simultaneously.

OPC UA (Open Platform Communications Unified Architecture) is an industrial M2M communication protocol based on AMPQ which can securely and reliably transport machine data (control variables, measured values, parameters) as well as semantically describe this data in a machine-readable way. It is a de facto standard in the automation field and is set for wide adoption in industrial IoT applications. It is most successful for Supervisory Control and Data Acquisition (SCADA) systems and is considered heavier than the publish/subscribe-oriented MQTT. However, OPC UA is enhanced in real time with a publisher/subscriber mechanism [56].

LWM2M (Lightweight M2M) is a messaging protocol designed by the Open Mobile Alliance (OMA) based on CoAP for managing IoT devices. It defines the interfaces for bootstrapping, device discovery, registration, and device management. It supports parameterization and monitoring of devices as well as the update of firmware. LWM2M also supports the communication between servers (nodes in a private or public network) and clients (embedded in a device).

As messaging protocol standards with different characteristics (message size, overhead, power consumption, bandwidth, reliability, security, etc.) proliferate, protocols need to be carefully evaluated in order to determine their fit for a particular IoT application [57].

1.3.4 Communication Platforms

In order for the protocols defined in the previous section (such as AMQP, MQTT, and XMPP) to be implemented in practice, it is necessary to define the procedures through which many IoT devices can communicate with other devices or with servers in an organized manner. Most communication platforms use a broker model, which provides trusted message prioritization, routing, filtering and delivery between a publisher and a subscriber [50]. Platforms can be configured as a centralized broker system (with all messages going through one server), a centralized multibroker system (with several brokers on different servers, each with its own message queue), or a decentralized broker system (which has no central server and uses the IP multicast protocol coupled with local client-side functionality for persistence, security, and transactions) [50]. We present examples of such brokers below.

HiveMQ is an Enterprise MQTT Broker conceptualized for commercial use, developed in Germany by dc-square GmbH. The broker supports the publish/subscribe MQTT standard. It has a cluster functionality to scale HiveMQ deployments (supporting millions of MQTT clients on different machines) in a horizontal linear manner [58]. Developers can implement their own mechanisms for cluster discovery.

An alternative for HiveMQ is the open-source implementation Mosquitto—an open-source MQTT broker. Mosquitto is written in C and is a project of the Eclipse IoT Working Group, which includes Bosch, Red Hat, and SAP [59]. Mosquitto also runs on a Raspberry Pi device.

Another open-source MQTT broker written in Java is Moquette. Like Mosquitto, it also runs on IoT devices such as on a Raspberry Pi [60]. Usually, Moquette runs stand-alone. However, it can also be integrated into an OSGi container like Concierge [61] or Kura [62].

Another alternative is Pivotal RabbitMQ, an open-source message broker. While not developed specifically for MQTT, it has an MQTT adapter, thus offering a full-fledged open-source MQTT broker as well [63].

The Eclipse Paho project provides open-source client implementations of MQTT and MQTT-SN messaging protocols which are aligned with the new, existing, and future IoT applications. The word Paho comes from the Maori language and means to distribute or send. Paho is a broadcast messaging protocol for IoT. An MQTT solution is often used by means of Paho (publisher and subscriber on client side) and Mosquitto (broker on server side). Paho is set up as a standard messaging library in the OSGi container Eclipse Kura [64].

1.3.5 Device Control, Integration, and Simulation Frameworks

IoT solutions usually involve a diversity of IoT devices interacting with each other and transmitting data to IoT-based applications. The device control, integration, and simulation frameworks enable easy configuration of these devices and efficient management of their operation. We present examples of such frameworks below.

The Java-based framework openHAB (open Home Automation Bus) integrates buildings automation components from different manufacturers in a single device-independent and protocol-independent platform [65]. openHAB is continuously adding bindings (which describe how to transport data in and out of a device) for end devices, such as Xiaomi Mi smart home devices, IKEA Trådfri smart lighting devices, and Gardena’s smart garden robotic lawn mowers [66, 67]. The technology-specific bindings communicate internally with the openHABCore via the openHAB event bus. However, the external communication of openHAB with the outside world takes place via an MQTT broker such as HiveMQ [66]. OpenHAB builds on the Eclipse SmartHome project [66, 68] and is also connected to the Eclipse IoT Market, where developers can find a connection to devices which are not included in openHAB (if there are no open-source implementation or suitable licenses) [66]. There is also a self-configuring Raspberry Pi setup called openHABian, which starts with an SD card image and automatically installs Java, openHAB, Samba, and other software. Industry giants like Z-Wave, KNX, Homematic, EnOcean and Insteon are already integrating their smart home products (such as smart lights, thermostats, alarm systems, etc.) with openHAB in order to ensure smooth connectivity of their various devices in smart home applications.

The Eclipse Edje project falls under the Apache license 2.0 and provides a hardware abstraction Java API (application programming interface) for accessing hardware features of microcontrollers such as the general-purpose input/output (GPIO) interfaces [69]. As the smallest common point of Java SE, Java SE Embedded, MicroEJ and Android, the Edje Device Configuration (EDC) provides a minimal execution environment (which covers Java standard packages java.lang, java.util and java.io) for a device compatible with Edje. Minimum requirements are a 32-bit processor with a frequency of 16 MHz, 32 KB RAM and a 128 KB flash memory [69].

Eclipse Ditto [70] provides access to digital twins (software abstractions of real-world devices) and mediates between the physical world and the digital twin representations. Digital twin is an IIoT solution that can be useful in understanding past and current performance of real-world machines, optimizing the operation of the machines, and predicting future operation and maintenance needs. Core aspects of the Eclipse Ditto framework include device-as-a-service (a higher abstraction level of the device in the form of an API, which is used to work with this device), state management of digital twins (in terms of current state data as well as configuration properties), and organization of the digital twins (geographic assignment and tracking of their geographical relationship, metadata assignment, and search functions) [70].

1.3.6 Tools and Frameworks for Modeling, Development, and Deployment

The development life cycle of IoT applications (specifying, modeling and developing the system, and deploying it in production mode) is supported by specific tools and frameworks, which we review below.

Eclipse 4diac is an Open-Source PLC (Programmable Logic Controller) platform for distributed industrial automation and management systems that implements the IEC 61499 standard [71]. The standard defines a domain-specific modeling language (DSL) for such systems. The platform includes four main components: a runtime environment (RTE), a development environment (IDE), a function block library (LIB), and sample system projects (SYS). The latter includes implementation of Intelligent Electronic Devices (IED) for smart grids which comply with IEC 61499 (model for distributed management systems) and with IEC 61850 (transfer protocol for protection and control technology in electrical switchboards of medium- and high-voltage technology for power supply system automation) [71].

Eclipse Vorto is an open-source tool for creating and managing abstract device descriptions which are neutral in terms of technology (i.e., so-called information models) [72, 73]. Vorto is operated by Bosch and as part of the Bosch Software Innovations (Bosch-SI) Group. It contributes to the standardization of information models for IoT devices. Device manufacturers can create these models with Eclipse Vorto by means of a text-based DSL editor which provides auto-complete, syntax highlighting and content assist (code hinting). The models are abstractions of real devices in terms of status, attributes, and functionalities; they should facilitate the communication between these devices.

Normally, the modeling is done by the device manufacturers. A meta-information model along with Eclipse-based tooling has been developed for designing new information models. Vorto provides a server-based repository for management, release, and reuse as well as collaborative work on these models. Diverse code generators create solutions for various environments such as for Eclipse SmartHome, openHAB, OSGi-DAL, Bosch or Kura. These code generators are provided by the environment providers.

There are numerous other Eclipse IoT projects [74]. Some of these projects include relevant tools, which have been incorporated in Eclipse's Open IoT Stack for Java [75].

Eclipse hawkBit is a *domain-independent back-end framework for rolling out software updates to constrained edge devices as well as more powerful controllers and gateways connected to IP based networking infrastructure* [76]. This industrial provisioning system is built on the Spring Boot framework, which can be adapted to any cloud platform. It also supports flexible deployment management for the rollout of updates on a massive number of devices, clustered as per separate deployment groups including, emergency shutdown, and progress monitoring for the entire rollout and for each group [76].

Papyrus for IoT is a modeling solution based on Eclipse Papyrus and is a part of S3P (Smart, Safe, and Secure Software Development and Execution Platform for IoT) research and development project. Papyrus for IoT enables specification, design, deployment, and monitoring of IoT systems and generates code for Vortex from PrismTech and for IoT-device-operating system from MicroEJ [74, 77].

SensIDL is an open-source development framework that helps sensor developers to specify and implement communication interfaces of intelligent sensors. The specification is done in the same way as Vorto via a DSL [74, 78]. SensIDL uses DSL as standard specification language for defining data provided by sensors, where this description serves as a basis for an automated code generation for the sensors as well as for the recipients. The generator integrated in SensIDL uses the interface definition as input to generate a semantically enriched interface-specific API [79].

Reactive Blocks is a tool which facilitates the development of software for IoT gateways using graphical modeling and code generation. The generated software is based on Java and OSGi [74, 80].

Node-RED is an open-source tool for graphical modeling and execution of processes in IoT applications—so-called flows. It serves to connect devices, APIs, and online services with each other. The modeled flows are interpreted for runtime. The tool is completely based on Node.js [74, 81].

Virtual Developer is a platform based on Eclipse for automation of programming tasks through code generators. It serves for the development of IoT applications by means of model-driven software development and generates software code for sensors and actors. It also has a Cloud Connector that sends models to servers and receives the generated code [82].

Resin.io is a container-based platform to develop, deploy, and manage code on IoT devices [83]. The platform serves for developing and deploying applications such as environment monitoring, customized retail experiences, or drone deployment on various devices (like Raspberry Pi, Arduino, etc.). Its features include fast iterative development, lightweight and reliable deployment, and management of devices, phased deployments, scheduled updates, etc. [83].

Eclipse Concierge is a lightweight implementation of the OSGi Core specification (a framework for modular software development and deployment), optimized for mobile and embedded devices [61, 84]. Thus, Concierge joins the OSGi framework alongside Equinox, Apache Felix, and Knopflerfish. Concierge has a low footprint; it provides itself as resources to the framework and its internal statuses via a REST (REpresentational State Transfer) interface. Concierge can be integrated with Eclipse SmartHome and OSGi enRoute [61, 84].

The Eclipse Hono project, supported by Bosch and Red Hat, provides a platform for scalable messaging in the field of IoT, where AMQP middleware is connected between devices and back-end services [85, 86]. Devices not supported by AMQP are connected with the protocol adapter, which is responsible for the conversion of AMQP and non-AMQP messages. Moreover, Eclipse Hono collects and processes telemetry data from remote devices centrally by enabling devices to send data to the messaging cloud. Additionally, back-end services can store telemetric data and send commands to the devices.

Eclipse Milo is an implementation of the OPC UA standard, which plays an important role in the automation industry [87]. Therefore, Milo supports IIoT needs, such as access to real-time data, monitoring alarms and events, access to historical data and data modeling.

Californium [88] is a CoAP framework focused on back-end services and IoT devices. It consists of five sub-projects. The Californium-core provides the central framework for protocol implementation in order to develop IoT applications. The sub-project Scandium provides security for Californium and is a pure Java implementation of Datagram Transport Layer Security (DTLS), an encryption protocol [89]. Actinium is the app-server for Californium for implementing IoT mashups.

The sub-project Connector abstracts from the various types of transport which can use CoAP. The sub-project CoAP tools includes tools such as a browser, a command line client, and the CoAPBench for benchmarking customized (CoAP-based) IoT applications [88].

Leshan depends on Californium and presents a complete infrastructure for Java and LWM2M-based IoT solutions. This includes libraries for server and client-side device management, a device management server with web interface as well as a bootstrapping server which is responsible for the security setup of the connected devices [90, 91].

Eclipse Kura is an OSGi-based container for M2M applications which enhances the OSGi and Java standard platforms with APIs and services, which are customized based on the requirements of M2M applications such as I/O access, data services, network configuration and telemetry [62, 92]. Kura is not installed on sensors, but on stronger devices (e.g., Raspberry PI), which function as IoT gateway (receiver/ingestor and distributor). Therefore, Kura undertakes the role of an IoT gateway, i.e., it is responsible for collecting messages from IoT devices and for processing, aggregating, and forwarding these messages. The rule-based routing and mediation engine Apache Camel is set up in Kura, which is responsible for orchestrating the flow of messages. The setup of Camel in Kura is very useful as Camel already includes about 200 OSGi-enabled connectors including JMS, REST, CoAP, AMQP, MQTT, etc.; it also has client-side load balancing. A Camel OSGi Bundle is started from Kura [62, 92]. Kura also contains an Eclipse-based development environment, in which the M2M applications can be developed in emulators and can then be deployed in a target gateway and subsequently transferred on the real-end devices in the field [92].

Eclipse SCADA (also called as Eclipse NeoSCADA) is a modular kit (including extensive IDE) used by developers to implement their own SCADA solutions. The kit includes different protocol adapters (including MQTT, REST, JDBC, Modbus, Siemens S7) and middleware, in order to process data from devices as well as some modules to cover basic functions of a SCADA system (e.g., alarms and events, recording historical data). It also includes user interface components, libraries, and a configuration framework [93, 94].

Eclipse OM2M is an open-source service platform for the interoperability of M2M based on the one M2M standard. OM2M follows a REST-compliant approach with open interfaces in order to facilitate the development of services and applications irrespective of the underlying network. The platform provides a modular architecture above the OSGi layer, which can be enhanced through plug-ins. It supports several protocol connections like HTTP and CoAP. Various proxies interacting smoothly enable seamless communication with manufacturer-specific technologies such as ZigBee [95].

Eclipse Wakaama, like Leshan, is an LWM2M implementation for the client or device side. However, it offers a C-based implementation (and not a Java library), which is portable on POSIX-compliant systems [96].

Temboo is a cloud-based platform with complete software stack for development of IoT applications through code generation. It provides generated code fragments, which a developer adds in his program code with "Copy & Paste" and processes it further. The code fragments support devices, APIs, databases and online services, where the generated code is dependent on the Temboo SDKs, which are available for different programming languages [74, 97].

Eclipse Ponte is an M2M Bridge Framework for REST development [98]. It serves for the creation of a reusable solution for bridging multiple M2M protocols as per REST. This means that developers formulate a REST-API to read, write and access the history of sensors, actors and other IoT devices. In addition, developers release the MQTT and/or CoAP messages outside using a REST-API. They define an internal API for easy addition of new protocols via plug-ins [98].

1.3.7 IoT Cloud Integration Platforms

Several cloud providers offer IoT integration platforms to facilitate the collection, processing, and analysis of IoT data using scalable computing. We review the main platforms below.

AWS IoT is the Amazon cloud platform that allows connected things to collaborate easily and securely with cloud applications and other devices. This platform is designed to reliably and securely support billions of connected devices and process trillions of messages, forwarding them to AWS end points or other smart devices. AWS IoT also provides a separate MQTT Message Broker and MQTT Client [99].

Microsoft provides an equivalent cloud platform called Azure IoT, which functions as a hub for reliable and secure bidirectional communication between millions of IoT devices and a solution back end. It also includes an IoT Suite end-to-end implementation of this communication architecture for specific IoT scenarios along with remote monitoring of device status, predictive maintenance and connected factory features for industrial installations [100].

IBM Watson IoT services facilitate easy and efficient application access to IoT devices and data, as well as real-time monitoring and analysis. These also enable the easy building of analytic applications, visualization dashboards, and mobile IoT applications [101].

1.4 Conclusions

In this chapter, we summarized and analyzed the components necessary for an integrated end-to-end IoT solution and highlighted specific IIoT components as appropriate. Our analysis goes beyond existing studies, including our own previous work, by specifically focusing on all IoT solution components—rather than just subsets—relating to both solution development and operations. To our knowledge, this is the first attempt to provide such an examination of the state of the art. By highlighting the advantages and disadvantages of choosing different IoT components, we hope that our analysis is helpful to IoT practitioners designing IoT systems with diverse requirements. We also hope that this analysis can help students as well who are interested in learning about IoT, as well as researchers interested in understanding the current IoT landscape and its limitations, to support them in the creation of new standards, tools, and frameworks.

Acknowledgements The authors would like to thank Capgemini for supporting the editing of the present contribution. The authors would also like to thank all the anonymous reviewers who made many valuable suggestions for improving the text.

References

1. Ashton K (2011) That 'internet of things' thing. RFID J 22
2. Columbus L (2018) A roundup of 2018 enterprise Internet of things forecasts and market estimates. Enterprise CIO
3. Gold J (2018) What is the industrial IoT? [and why the stakes are so high]. Network World
4. Hofmann E, Rüsch M (2017) Industry 4.0 and the current status as well as future prospects on logistics. Comput Ind 89:23–34
5. Columbus L (2018) 10 charts that will challenge your perspective of IoT's growth. Forbes
6. Papert M, Pflaum A (2017) Development of an ecosystem model for the realization of internet of things (IoT) services in supply chain management. Electron Markets 27:175–189
7. Prasse C, Nettstraeter A, ten Hompel M (2014) How IoT will change the design and operation of logistics systems. In: Proceedings of IEEE international conference on the internet of things (IOT). Cambridge, MA, USA
8. Guerrero-ibanez JA, Zeadally S, Contreras-Castillo J (2015) Integration challenges of intelligent transportation systems with connected vehicle cloud computing, and internet of things technologies. IEEE Wireless Commun 22:122–128
9. He W, Yan G, Xu LD (2014) Developing vehicular data cloud services in the IoT environment. IEEE Trans Industr Inf 10:1587–1595
10. Iansiti M, Lakhani KR (2014) Digital ubiquity: how connections, sensors, and data are revolutionizing business. Harvard Bus Rev 92:91–99
11. Chen S, Xu H, Liu D, Hu B, Wang H (2014) A vision of IoT: applications challenges, and opportunities with China perspective. IEEE Internet of things J 4:349–359
12. Kim J, Yun J, Choi SC, Seed DN, Lu G, Bauer M, Al-Hezmi A, Campowsky K, Song J (2016) Standard-based IoT platforms interworking: implementation, experiences, and lessons learned. IEEE Commun Mag 54:48–54
13. Chen C, Helal S (2008) Sifting through the jungle of sensor standards. IEEE Pervasive Comput 7:84–88
14. Chircu AM, Sultanow E, Sözer L (2017) A reference architecture for digitalization in the pharmaceutical industry. In: Eibl M, Gaedke M (eds) Workshops der INFORMATIK 2017. Lecture notes in informatics (LNI). Gesellschaft für Informatik, Bonn
15. Sultanow E, Chircu AM, Schroeder K, Kern S (2018) A reference architecture for pharma, healthcare & life sciences: a framework for using digital technology. In: Czarnecki C et al (eds) Workshops der INFORMATIK 2018. Lecture Notes in Informatics (LNI). Gesellschaft für Informatik, Bonn
16. Sultanow E, Chircu AM (2018) Bringing clarity to the java IoT jungle. Issues Inform Syst 19(4):26–34
17. Boyes H, Hallaq B, Cunningham J, Watson T (2018) The industrial Internet of things (IIoT): an analysis framework. Comput Ind 101:1–12

18. Lasi H, Fettke P, Kemper HG, Feld T, Hoffman M (2014) Industry 4.0, business & information. Syst Eng 6:239–242
19. Drath R Horch A (2014) Industrie 4.0: hit or hype? [industry forum]. IEEE Industrial Electron Mag 8(2):56–58
20. Lu Y (2017) Industry 4.0: a survey on technologies, applications and open research issues. J Industrial Inf Integr 6:1–10
21. Lin J, Yu W, Zhang N, Yang X, Zhang H, Zhao W (2017) A survey on internet of things: architecture enabling technologies, security and privacy, and applications. IEEE Internet of things J 4(5):1125–1142
22. Xu LD, He W, Li S (2014) Internet of things in industries: a survey. IEEE Trans Industr Inf 10(4):2233–2243
23. Pang Z, Zheng L, Tian J, Kao-Walter S, Dubrova E, Chen Q (2015) Design of a terminal solution for integration of in-home health care devices and services towards the internet-of-things. Enterp Inf Syst 9(1):86–116
24. Ramirez ARG, González-Carrasco I, Jasper GH, Lopez AL, Lopez-Cuadrado JL, García-Crespo A (2017) Towards human smart cities: internet of things for sensory impaired individuals. Computing 99(1):107–126
25. Dhariwal K, Mehta A (2017) Architecture and plan of smart hospital based on internet of things (IOT). Int Res J Eng Technol 4(4):1976–1980
26. Zhang Y, Zhang G, Wang J, Sun S, Si S, Yang T (2015) Real-time information capturing and integration framework of the internet of manufacturing things. Int J Comput Integr Manuf 28(8):811–822
27. Zancul EDS, Takey SM, Barquet APB, Kuwabara LH, Cauchick Miguel PA, Rozenfeld H (2016) Business process support for IoT based product-service systems (PSS). Bus Process Manage J 22(2):305–323
28. Wang K, Wang Y, Sun Y, Guo S, Wu J (2016) Green industrial internet of things architecture: an energy-efficient perspective. IEEE Commun Mag 54(12):48–54
29. Liu X, Zhao M, Li S, Zhang F, Trappe W (2017) A security framework for the internet of things in the future internet architecture. Future Internet 9(3):1–28
30. Yang Y, Wu L, Yin G, Li L, Zhao H (2017) A survey on security and privacy issues in internet-of-things. IEEE Internet of things J 4(5):1250–1258
31. Lin SW, Murphy B, Clauer E, Loewen U, Neubert R, Bachmann G, Pai M, Hankel M (2017) Architecture alignment and interoperability: an industrial internet consortium and platform industries 4.0 joint whitepaper. Industrial Internet Consortium, 5 Dec 2017, https://www.iiconsortium.org/pdf/JTG2_Whitepaper_final_20171205.pdf. Accessed 25 Jan 2019
32. IBM (2018) Internet of things for Insights from Connected Devices, https://www.ibm.com/cloud/garage/architectures/iotArchitecture. Accessed 25 Jan 2019
33. Object Management Group (2018) DDS, https://www.omgwiki.org/dds/. Accessed 25 Jan 2019
34. Open Connectivity Foundation (2018) Solving The IoT Standards Gap, https://openconnectivity.org/. Accessed 25 Jan 2019
35. Internet of things Consortium (2018) Internet of things Consortium (IoTC), https://www.iotone.com/organization/internet-of-things-consortium-iotc/o182. Accessed 25 Jan 2019
36. Eclipse Foundation (2018) Open Source for IoT, https://iot.eclipse.org/. Accessed 25 Jan 2019
37. Singh K, Kapoor D (2017) Create your own internet of things: a survey of IoT platforms. IEEE Consum Electron Mag 6:57–68
38. CNXSoft (2016) How to use Sonoff POW ESP8266 WiFi power switch with MQTT and Thing Speak, 11 Dec 2016, https://www.cnx-software.com/2016/12/11/how-to-use-sonoff-pow-esp8266-wifi-power-switch-with-mqtt-and-thingspeak/. 17 Nov 2018
39. Gold J (2016) Sigfox and LoRa are the WiMax of IoT. Computerworld, 12 Sep 2016, https://www.computerworld.com/article/3117795/cloud-computing/sigfox-and-lora-are-the-wimax-of-iot.html. Accessed 17 Nov 2018
40. Deutsche Telekom (2017) Narrowband IoT: the game changer for the internet of things. 1 Oct 2017, http://m2m.telekom.com/fileadmin/media/Whitepaper_NarrowBand_IoT_-_The_Game_Changer_for_the_Internet_of_Things_-_1.10.2017.pdf. Accessed 17 Nov 2018

41. Hwang Y (2018) Cellular IoT explained—NB-IoT versus, LTE-M versus, 5G and More, https://www.leverege.com/blogpost/cellular-iot-explained-nb-iot-vs-lte-m. Accessed 29 Feb 2018
42. Mehboob U, Zaib Q, Usama C (2016) Survey of IoT communication protocols techniques, applications, and issues, http://xflowresearch.com/wp-content/uploads/2016/02/Survey-of-IoT-Communication-Protocols.pdf. Accessed 25 Feb 2018
43. Viswanathan P (2018) 4G mobile networks: the pros and the cons. 12 Nov 2018, https://www.lifewire.com/4g-mobile-networks-pros-and-the-cons-2373260. Accessed 17 Nov 2018
44. Hwang Y (2016) Cellular IoT explained—NB-IoT versus, LTE-M versus, 5G and More, 30 Dec 2016, https://www.leverege.com/blogpost/cellular-iot-explained-nb-iot-vs-lte-m. Accessed 25 Feb 2018
45. Matten L (2016) NB-IoT: pros and cons of the new LPWA radio technology, 11 Oct 2016, https://de.slideshare.net/M2M_Alliance/nbiot-pros-and-cons-of-the-new-lpwa-radio-technology. Accessed 25 Feb 2018
46. Obermaier D (2015) IoT-Protokolldschungel—Ein Wegweiser, 17 Nov 2015, https://www.informatik-aktuell.de/betrieb/netzwerke/iot-protokolldschungel-ein-wegweiser.html. Accessed 17 Nov 2018
47. Stanford-Clark A, Truong HL (2013) MQTT for sensor networks (MQTT-SN) protocol specification Version 1.2. International Business Machines Corporation (IBM), 14 Nov 2013, http://mqtt.org/new/wp-content/uploads/2009/06/MQTT-SN_spec_v1.2.pdf
48. Diwan M, D'Souza M (2017) A framework for modeling and verifying IoT communication protocols. In: Larsen K, Sokolsky O, and Wang J (eds) Dependable software engineering. Theories, tools, and applications. SETTA 2017. Lecture Notes in Computer Science. Springer, Cham
49. Prado J (2016) OMA lightweight M2M resource model, https://www.iab.org/wp-content/IAB-uploads/2016/03/OMA_LightweighM2M_Resource_Model_Summary.pdf. Accessed 25 Feb 2018
50. PrismTech (2017) Messaging technologies for the industrial internet and the internet of things whitepaper. 12 May 2017, http://www.prismtech.com/sites/default/files/documents/Messaging-Whitepaper-051217.pdf. Accessed 25 Feb 2018
51. Melo M (2018) CoAP and MQTT-SN: explained, https://www.sine-wave.com/blog/mqtt-sn-and-coap#.WpLrqExFzvM. Accessed 25 Feb 2018
52. Semle A (2015) IIoT protocols to watch, 26 Oct 2015, https://www.automation.com/library/white-papers/iiot-protocols-to-watch. Accessed 25 Feb 2018
53. Kowalke M (2015) The pros and cons of the major IoT communications protocols, 20 Aug 2015, http://www.realtimecommunicationsworld.com/topics/realtimecommunicationsworld/articles/408622-pros-cons-the-major-iot-communications-protocols.htm. Accessed 25 Feb 2018
54. GitHub (2018) XMPP-IoT, https://github.com/joachimlindborg/XMPP-IoT. Accessed 17 Nov 2018
55. Rahman RA, Babar S (2017) Security analysis of IoT protocols: a focus in CoAP. In: Proceedings of the 3rd MEC international conference on big data and smart city (ICBDSC), Muscat, Oman, March 2016, pp 1–7
56. B&R Industrial Automation GmbH (2018) TSN and Pub/Sub: real-time capability for OPC UA, https://www.br-automation.com/en/technologies/opc-ua/tsn-and-pubsub/. Accessed 17 Nov 2018
57. Naik N (2017) Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. In: Proceedings IEEE international systems engineering symposium (ISSE), Vienna, Austria, Oct 2017, 1–7
58. Feldmann M (2016) A new cluster for HiveMQ 3.1, 7 Mar 2016, https://jaxenter.de/ein-neuer-cluster-fuer-hivemq-3–1-36033. Accessed 17 Nov 2018
59. Eclipse Foundation (2018) Open Source for IoT, https://iot.eclipse.org/. Accessed 17 Nov 2018

60. Eclipse Foundation (2018) Moquette MQTT, https://projects.eclipse.org/projects/iot.moquette. Accessed 17 Nov 2018
61. Eclipse Foundation (2018) Eclipse Concierge, https://projects.eclipse.org/projects/rt.concierge. Accessed 17 Nov 2018
62. Eclipse Foundation (2018) Kura, https://www.eclipse.org/kura/. Accessed 17 Nov 2018
63. RabbitMQ (2018) MQTT Adapter, https://www.rabbitmq.com/mqtt.html. Accessed 17 Nov 2018
64. Eclipse Foundation (2018) Eclipse Paho, https://www.eclipse.org/paho/. Accessed 17 Nov 2018
65. Artmann M (2017) OpenHAB 2.1—new level for smart-home-management, 9 Sep 2017, https://www.homeandsmart.de/openhab-2-smart-home-software-open-source. Accessed 17 Nov 2018
66. Mann M, Götz C (2015) Smart home in action with openHAB and MQTT. Eclipse Magazin 2
67. Menge R (2017) Smart home: openHAB 2.1 can access Eclipse IoT Market, 28 Jun 2017, https://www.heise.de/developer/meldung/Smart-Home-openHAB-2–1-kann-auf-Eclipse-IoT-Market-zugreifen-3757723.html. Accessed 17 Nov 2018
68. Schmidt J (2014), Eclipse smarthome is designed to prevent fragmentation in the smart home area, 17 Jun 2014, https://www.heise.de/developer/meldung/Eclipse-SmartHome-soll-Fragmentierung-im-Smart-Home-Bereich-verhindern-2225118.html. Accessed 17 Nov 2018
69. Eclipse Foundation (2018) Edje, https://projects.eclipse.org/proposals/edje. Accessed 17 Nov 2018
70. Eclipse Foundation (2018) Eclipse Ditto, https://projects.eclipse.org/proposals/eclipse-ditto. Accessed 17 Nov 2018
71. Eclipse Foundation (2018) Eclipse 4diac, https://www.eclipse.org/4diac/. Accessed 17 Nov 2018
72. Eclipse Foundation (2018) Eclipse Vorto, https://www.eclipse.org/vorto/. Accessed 17 Nov 2018
73. Scheib J, Laverman J, Wagner M, Weinmann O (2016) Eclipse Vorto interoperability for Internet of things. Eclipse Magazin 3
74. Munzert M (2016) Industrial IoT solutions with eclipse IoT and model-driven development, 21 May 2016, http://www.informatik-aktuell.de/entwicklung/methoden/industrielle-iot-loesungen-mit-eclipse-iot-und-mdsd.html. Accessed 17 Nov 2018
75. Eclipse Foundation (2018) Open IoT stack for java, https://iot.eclipse.org/java/open-iot-stack-for-java.html. Accessed 17 Nov 2018
76. Eclipse Foundation (2018) hawkBit, https://projects.eclipse.org/proposals/hawkbit. Accessed 17 Nov 2018
77. Dhouib S, Cuccuru A, Le Fèvre F, Li S, Maggi B, Paez I (2016) Papyrus for IoT—a modeling solution for IoT. In: Proceedings l'Internet des Objets (IDO: Nouveaux Défis de l'Internet des Objets: Interaction Homme-Machine et Facteurs Humains. Paris, France
78. SensIDL (2018) A generic framework for implementing sensor communication interfaces, http://sensidl-project.github.io/SensIDL/. Accessed 17 Nov 2018
79. Groenda H, Rathfelder C, Taspolatoglu E (2015) SensIDL: Ein Werkzeug zur Vereinfachung der Schnittstellenimplementierung intelligenter Sensoren, https://www.sigs-datacom.de/uploads/tx_dmjournals/Groenda_Rathfelder_Taspolatoglu_OTS_IoT_2015.pdf. Accessed 17 Nov 2018
80. BitReactive (2018) Reactive Blocks, http://www.bitreactive.com/reactive-blocks. Accessed 17 Nov 2018
81. Node-RED (2018) Node-RED Flow-based programming for the Internet of things, https://nodered.org. Accessed 17 Nov 2018
82. Generative Software (2018) Virtual Developer, https://www.virtual-developer.com. Accessed 17 Nov 2018
83. Resin.io (2018) Resin.io, https://resin.io. Accessed 17 Feb 2018

84. Hiller J (2015) Eclipse concierge—fit for IoT? OSGi for Embedded Systems in the Internet of things. Eclipse Magazine, USA, p 2
85. Eclipse Foundation (2018) Eclipse Hono, https://projects.eclipse.org/projects/iot.hono. Accessed 17 Nov 2018
86. Mohilo D (2016) Eclipse Weekly: Neon M7, Andmore-Update und das neue Projekt Eclipse Hono, 11 May 2016, https://jaxenter.de/eclipse-weekly-neon-m7-andmore-update-und-das-neue-projekt-eclipse-hono-40055. Accessed 17 Nov 2018
87. Eclipse Foundation (2018) Eclipse Milo, https://projects.eclipse.org/projects/iot.milo. Accessed 17 Nov 2018
88. Eclipse Foundation (2018) CoAP in Java, https://www.eclipse.org/californium/. Accessed 17 Nov 2018
89. Capossele A, Cervo V, De Cicco G, Petrioli C (2015) Security as a CoAP resource: an optimized DTLS implementation for the IoT. In: Proceedings of IEEE international conference on communications (ICC), London, UK
90. Schlosser H (2014) New IoT Project Proposed: Eclipse Leshan, 12 Sep 2014, https://jaxenter.de/neues-iot-projekt-vorgeschlagen-eclipse-leshan-639. Accessed 17 Nov 2018
91. Eclipse Foundation (2018) Leshan, http://www.eclipse.org/leshan/. Accessed 17 Nov 2018
92. Schlosser H (2013) Eclipse Kura: Internet of things on OSGi, 25 Jul 2013, https://jaxenter.de/eclipse-kura-das-internet-der-dinge-auf-osgi-2889. Accessed 17 Nov 2018
93. Rose J, Reimann J (2015) Tutorial for IoT project for Eclipse: Eclipse SCADA tutorial, 3 Mar 2015, https://jaxenter.de/eclipse-scada-tutorial-teil-1-15166. Accessed 17 Nov 2018
94. Eclipse Foundation (2018) Eclipse NeoSCADA, https://eclipse.org/eclipsescada/. Accessed 17 Nov 2018
95. Eclipse Foundation (2018) Eclipse OM2M, https://projects.eclipse.org/projects/technology.om2m. Accessed 17 Nov 2018
96. Eclipse Foundation (2018) Wakaama, https://eclipse.org/wakaama/. Accessed 17 Nov 2018
97. Temboo (2018) Tools for digital transformation, https://temboo.com/. Accessed 17 Nov 2018
98. Eclipse Foundation (2018) Eclipse Ponte. https://projects.eclipse.org/projects/technology.ponte. Accessed 17 Nov 2018
99. Amazon (2018) AWS IoT https://aws.amazon.com//iot. Accessed 17 Nov 2018
100. Microsoft (2018) Overview of the Azure IoT *Hub Service*, https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-what-is-iot-hub#iot-device-connectivity-challenges. Accessed 19 Dec 2018
101. IBM (2018) Watson Internet of things, https://www.ibm.com/internet-of-things. Accessed 19 Dec 2018