A 360-DEGREE VIEW OF IoT TECHNOLOGIES

John Soldatos

A 360-Degree View of IoT Technologies

For a listing of recent titles in the
Artech House Integrated Microsystems Series,
turn to the back of this book.

A 360-Degree View of IoT Technologies

John Soldatos

ARTECH HOUSE
BOSTON | LONDON
artechhouse.com

Library of Congress Cataloging-in-Publication Data
A catalog record for this book is available from the U.S. Library of Congress.

British Library Cataloguing in Publication Data
A catalog record for this book is available from the British Library.

ISBN-13: 978-1-63081-752-7

Cover design by Andy Meaden, meadencreative.com

685 Canton Street
Norwood, MA 02062

10 9 8 7 6 5 4 3 2 1

Contents

CHAPTER 7

CHAPTER 8

CHAPTER 9

CHAPTER 10

Preface

The vision of the Internet of Things (IoT) was introduced more than 20 years ago. It was initially validated based on technologies such as sensors, wireless sensor networks (WSN), and radio frequency identification (RFID). Over the years, IoT has gradually moved from the realm of research experimentation to enterprise deployment. Early small-scale IoT systems gave way to massively scalable IoT applications that are integrated in the cloud and comprise different types of internet-connected objects. In recent years, novel architectural paradigms such as edge computing have enabled high-performance, real-time applications that collect and process large volumes of data close to the field. Furthermore, state-of-the-art IoT deployments are no longer composed of sensors and passive devices only. They also include novel cyber-physical systems that interact with the field and influence the status of the physical world. This is the case with smart objects such as drones, industrial robots, and automated guided vehicles.

Currently, IoT has a growing momentum, which is reflected in the rapid proliferation of the number and types of internet-connected devices. Beyond advances in IoT devices, IoT's growth is propelled by the accelerated growth of advanced digital technologies such as big data and artificial intelligence. These technologies enable more versatile and intelligent applications. Furthermore, emerging networking technologies (e.g., fifth generation (5G) communication infrastructures) are developed to boost the scalability, performance, and quality of service (QoS) of emerging IoT applications such as connected cars and autonomous driving. Future IoT systems will be based on the integration of a pool of advanced IoT-related technologies, including device, networking, data processing, data analytics, and application development technologies.

IoT is already disrupting entire sectors, such as trade, retail, logistics, healthcare, and industry. IoT applications in these sectors are driving significant improvements in the automation, accuracy, and cost efficiency of business processes. Moreover, IoT innovators are currently presented with unprecedented opportunities for improving business competitiveness and citizens' well-being. This was made evident during the recent COVID-19 pandemic outbreak, where many IoT tools were developed in just few weeks' time to fight the healthcare crises. Specifically, IoT applications for diagnostic testing, contact tracing, and disease spreading estimation were rapidly developed and used at very large scales. IoT applications are also playing an instrumental role in meeting some of the most important socioeconomic challenges of our time, such as climate change and urban safety. For example, IoT

technologies are widely used in applications that optimize natural resources and improve environmental performance in environments such as smart homes, smart buildings, and smart cities.

To understand and fully leverage IoT's disruptive potential, IoT stakeholders need insights in a broad range of IoT technologies and their role in IoT applications. Acquiring such insights can be particularly challenging, given the variety of IoT systems, architectures, and standards. This book aims to facilitate IoT stakeholders to navigate and understand the complex landscape of IoT technologies. It is also designed to help them understand how the different technologies can be integrated in value-added applications. In this direction, this book presents the most popular and widely used IoT technologies, including legacy IoT technologies such as WSN and RFID, and state-of-the-art technologies such as big data and cloud or edge computing. It also highlights future trends in the integration of IoT with emerging technologies such as 5G, distributed ledger technologies (i.e., blockchains), and artificial intelligence.

The rising popularity of IoT technologies and applications has led many authors to write books about IoT, such as guides for developers and architects, and books about research and scientific aspects of the IoT paradigm. Many of these books are excellent reads for researchers, practitioners, and managers who wish to understand the structure and underlying technologies of nontrivial IoT systems. Nevertheless, most of these books develop focused, yet quite narrow views of IoT technology. For example, some emphasize the devices' part of an IoT system, while others are devoted to networking technologies and the processing of IoT data with proper QoS. In this book, we aim to provide a more complete and comprehensive view of both legacy and modern IoT systems (i.e., a 360° view of IoT technologies and applications). Hence, this book delves into the details of a rich set of individual IoT technologies while emphasizing security issues. It also presents popular IoT standards and emerging IoT technologies.

The target audience for the book includes students, researchers, professionals, and practitioners:

- For students and researchers, this book can be a complete and comprehensive introduction to IoT technologies (i.e., it can serve as the first step in their IoT research and engineering endeavors). After reading the book, they will be ready to delve into more details on selected technologies such as big data, 5G, and artificial intelligence.
- For IoT engineers and professionals, this book can shed light on the building blocks that comprise a practical IoT deployment. It can introduce them to the different technology options that they have in order to build, deploy, and operate a nontrivial IoT system that solves real business problems.

Writing this book was the result of work on IoT projects and technologies for a period of over 15 years. In the scope of this journey, I had the chance to meet and collaborate with a cohort of excellent IoT researchers, engineers, and practitioners. With most of them, we collaborated in the scope of European research projects such as the Open Source cloud solution for the Internet of Things (OpenIoT)

project. With others, we worked together in various working groups of the European Research Cluster on the Internet of Things (IERC) and, recently, the Alliance for Internet of Things Innovation (AIOTI). I am truly grateful to these colleagues and acknowledge their help in acquiring some of the insights of this book.

I especially thank Dr. Ing. Panos Dimitropoulos, the former technical director of SENSAP Microsystems and one of the most prominent IoT experts. His technical vision and deep knowledge of IoT technologies have always been drivers for innovative IoT projects and solutions. I also thank my colleague Nikos Kefalakis, with whom I collaborated on more than 10 different IoT projects. Our exchange of ideas on IoT systems and applications has always been a valuable source of inspiration. Nikos was the person leading most of our joint IoT implementation activities. I feel grateful for our collaboration and acknowledge his contribution to this book.

Despite the wealth of available books on IoT topics, I strongly believe that reading the present book is worthwhile. I hope that this book will be a valuable and interesting resource for the IoT community. I wish to IoT researchers and practitioners who have chosen this book a good reading experience.

CHAPTER 1

Introduction to the Internet of Things

This chapter introduces what the Internet of Things (IoT) is. To this end, it uses several definitions that have been provided by different organizations. The chapter illustrates the main concepts behind the IoT computing paradigm, along with its main drivers. Moreover, it presents some of the most popular IoT technologies and applications, including the ones that will be detailed in subsequent chapters. At the end of this chapter, examples of popular IoT devices are given. Specifically, devices commonly used in enterprise and educational settings are presented. These devices provide some concrete examples of what "things" look like in the scope of the IoT paradigm. Using these popular devices, researchers, developers, and users can do practical work with IoT and hence understand how IoT applications work in practice.

1.1 Defining IoT

IoT is not tied to a single technology or type of application. Hence, it has been always difficult to provide a precise description of it. There have been several definitions of the IoT, as several organizations have attempted to describe this novel computing paradigm. For example:

- One of the first definitions of IoT was given almost 20 years ago by Kevin Ashton, cofounder of the Auto-ID Center at the Massachusetts Institute of Technology (MIT): "System where the Internet is connected to the physical world via ubiquitous sensors." This early definition has been used by several organizations including standards development bodies such as the Organization for the Advancement of Structured Information Standards (OASIS).
- The International Telecommunication Union (ITU) defined IoT as [1]: "a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies."
- The European Technology Platform on Smart Systems Integration (ETP EPoSS) has defined IoT as: "Things having identities and virtual personalities operating in smart spaces using intelligent interfaces to connect and

communicate within social, environmental, and user contexts." (https://iot.ieee.org/images/files/pdf/IEEE_IoT_Towards_Definition_Internet_of_Things_Issue1_14MAY15.pdf)

- These definitions reveal the complexity of IoT computing. In general, one can consider IoT as the computing paradigm that enables the combination of data and services from all the entities that are connected to the internet (i.e., all internet-connected objects) towards providing added-value services to citizens and businesses [2]. This definition will become clearer in the rest of the chapter, where we explain the main drivers behind the emergence and rise of the IoT.

Based on these definitions, one may wonder whether IoT is simply a rebranding of older concepts such as ubiquitous and pervasive computing. Indeed, nearly 30 years ago, Mark Weisser at the Computer Science Lab, Xerox PARC, identified three eras of computing:

- *Mainframe computers:* In this era, one (super) computer was used to serve multiple people through their terminal devices. This was typically how computers worked in the 1980s.
- *Personal computing:* In this era, every person used his or her own (desktop or laptop) personal computer.
- *Ubiquitous computing:* In this era, many computers provide services to one person. This is practically the present era, as it is now common for people to possess many devices (e.g., laptops, mobile phones, tablet computers). Moreover, most computing services are delivered based on more than one computing system including different types of pervasive computing systems such as smart sensors, robots, and wearable devices.

Overall, the vision of ubiquitous computing, as expressed three decades ago, included several of the concepts of the IoT computing paradigm. However, in the past decade, there has been a surge of interest in IoT for various reasons, as explained next.

1.2 Why IoT Now?

Several factors have driven the emergence and rise of IoT. Several of them have appeared intensified in the past few years, and this has significantly increased the relevance of the IoT paradigm.

1.2.1 Technology Acceleration

We are living in the era of technology acceleration, where the pace of technological progress speeds up exponentially over time. This is particularly true for information technology, which is growing exponentially and propelling accelerated change as

well. The technological acceleration is reflected on several factors that boost IoT's growth.

1.2.1.1 Proliferating Number of Devices

As part of the technological acceleration, the number of internet-connected devices is proliferating in an exponential pace. According to the IoT Analytics research firm, the number of internet-connected devices on the planet in 2019 exceeded 7 billion. This number did not account for devices such as tablets and smartphones. When considering smartphones and portable computers, the number of connected devices exceeds 17 billion. Furthermore, it is expected that the number of connected devices will continue to grow exponentially. According to recent research by Transforma Insights, the number of active IoT devices is expected to exceed 24 billion in 2030 [3]. The same study indicated a compound annual growth rate (CAGR) of 11% in the number of IoT devices. It should be also noted that the number of connected devices increases at a greater pace than the number of people on the planet. It was around 2005 when the number of connected smart phones exceeded the planet's population. In 2008, the number of IoT devices exceeded the number of people on the planet, even without counting smart phones and laptops. The point in time where the number of internet-connected devices exceeded the number of people on the planet is considered a major milestone in the evolution of the IoT paradigm.

1.2.1.2 Different Types of Devices

In addition to the exponential increase in the number of devices, we are also witnessing a proliferation of the different types of connected devices. Smart watches, Fitbits, smart textiles (e.g., clothing that can sense temperature and other parameters), boards that can host multiple sensors and actuators, connected thermometers, smart energy meters, internet-connected cameras, and pervasive sensors that can be attached to sports shoes are only some of the available internet-connected devices. These devices serve different purposes, measure different parameters, and are deployed in various IoT applications. It is nowadays common for consumers to purchase multiple devices of different types. At the same time, thousands of different devices are also deployed in industrial settings such as manufacturing shop floors, energy plants, and oil refineries.

Another factor that justifies the emergence and rise of IoT is that all these different devices are connected to the Internet. This is no longer the norm just for computers and smart phones, but also for a larger number of other devices such as coffee machines, cars, and electronic scales.

1.2.1.3 Moore's Law and the Exponential Increase in Computing Power

In 1965, Gordon Moore, the chief executive officer (CEO) of Intel, observed that the number of transistors in a dense integrated circuit doubles about every 2 years. At that time, nobody expected that this observation would go on for decades. Moore projected that this rate of growth would continue for at least another

decade. However, this exponential growth has been continuing to date, which facilitates the fast processing of large amounts of IoT data. The abundance of computing cycles is therefore one more factor that contributes to the rapid growth of IoT.

1.2.2 People Connect to Things

Another growth factor for IoT is that people can connect to things. As things are connecting to the internet, they can access information systems (e.g., a hospital information system or an enterprise resource planning (ERP) system), but also user devices (e.g., wearables and smart phones). Therefore, user devices enable human users to interact with other devices and facilitate things' interactions with people. As a prominent example, a person instrumented with sensors (e.g., motion sensors and electrocardiogram (ECG) sensors) can interact with a smart phone. Different interaction scenarios between a person and a device can be implemented in this context.

1.2.3 Things Connect to Things

Another IoT interaction scenario involves things connecting to things for the purpose of exchanging information or even enabling one system (thing) to invoke and take advantage of the services of the other (thing). Interactions between things are very common in the IoT paradigm. As a prominent example, a vehicle can interact with other vehicles as part of vehicle-to-vehicle (V2V) applications. Likewise, a vehicle can interact with the infrastructure of the city, which is usually referred to as vehicle-to-infrastructure (V2I). V2V and V2I are among the most common IoT applications in smart transport. The interactions between the various things are enabled by complex and heterogeneous networks that facilitate the connectivity of the various devices [4].

1.2.4 Climate Change

Climate change is one of the most important phenomena of our age. It refers to a long-term shift in average weather patterns, which results in significant and abnormal variations in weather conditions. To fight climate change, governments and enterprises all around the world design and implement "green" projects that improve environmental performance. Several IoT applications hold the promise to boost environmental performance and green friendliness, such as projects that boost the greater use of renewables. Such projects increase the demand for IoT applications and propel IoT's growth.

1.2.5 The Rise of the Fourth Industrial Revolution (Industry 4.0)

Technological revolutions are typically characterized by the rapid introduction of new technologies that replace legacy ones in short amounts of time. An industrial

revolution is taking placing during the past few years, based on the introduction of IoT, artificial intelligence (AI) and other digital technologies [5]. These technologies enable the digitalization of physical processes and drive rapid change in industrial sectors such as manufacturing, energy, oil and gas, mining, and healthcare. This industrial revolution is characterized as the fourth industrial revolution (Industry 4.0). Industry 4.0 blurs the boundaries between the digital, physical, and biological worlds. IoT is playing a key role in the fourth industrial revolution, which also explains the surge of interest in IoT technologies and applications.

1.3 IoT Versus Related Technologies

IoT is very closely related to various other technological concepts, which involve the integration and coordination of data and services from multiple connected devices. Some of these concepts are used interchangeably with IoT. However, in most cases, these technological terms and buzzwords have either subtle or bigger differences to the IoT paradigm. Let us explore some of the terms that are very relevant to IoT and are, in several cases, used instead of it.

1.3.1 Ubiquitous Sensor Networks (USN)

USN refers to applications that are empowered by large sets of connected sensors, which are, in most cases, distributed both geographically and administratively. USN can be considered as a subset of IoT in the sense that it considers the interconnection and coordination of multiple sensing devices. However, IoT is not limited to sensing devices and therefore IoT has a much broader scope than USN.

1.3.2 Machine-to-Machine (M2M) Communications

M2M is a very commonly used term among telecommunications organizations, which use it to denote things-to-things interactions. As already outlined, M2M or things-to-things interactions are supported by IoT. However, IoT supports additional interactions as well and hence it is a superset of M2M. Nevertheless, M2M applications are certainly a very important segment of IoT.

1.3.3 Internet of Everything (IoE)

IoE is another term used instead of IoT. It has been propelled by IoT vendors such as Cisco. IoE refers to the networked connection of things, people, data, and processes, and, as such, it is sometimes considered broader than IoT. Nevertheless, the scope of IoT includes the connection of devices, people, and processes as well. Hence, IoT can be used interchangeably with IoE. The choice between the two terms is usually based on marketing and branding criteria, rather on the different capabilities and functionalities implied by each one of the two.

1.3.4 Cloud of Things (CoT)

CoT is common in cases where IoT data and services are deployed and/or used in the cloud. It is a term inspired by the very close affiliation between IoT and cloud computing. Nevertheless, IoT applications are not, in all cases, cloud-based, which makes CoT a subset of IoT.

1.3.5 Web of Things (WoT)

WoT refers to systems that converge web concepts with IoT concepts towards enabling end users or developers to access IoT data and IoT services over the web [6, 7]. The emergence of the WoT concept has its roots in that many IoT applications are web-based. It should be noted that not all IoT applications are web-based, which makes WoT a subset of IoT rather than a synonym for it.

1.3.6 Cyber-Physical Systems (CPS)

CPS refers to networked systems that combine information and services from both the physical world and the cyber world. Moreover, CPS use information from the digital world to drive physical processes and the operation of physical parts of the system. It is therefore evident that CPS functionality is covered by IoT definitions. In practice, CPS is used instead of IoT in cases of systems that comprise physical systems and processes, while at the same time involving actuation and control. This is the case in several application areas such as smart manufacturing in the scope of the fourth industrial revolution (Industry 4.0), which is enabled by CPS.

One will certainly listen, read, and use some of these terms instead of IoT. It is good to use the proper term in the proper context, which requires an understanding of the scope and the meaning of each of these terms.

1.4 IoT Technologies and Applications

1.4.1 IoT Technologies

IoT is not a single technology, which is why its definition is quite complex and comes in many different flavors. Rather, IoT is an entire paradigm that is empowered by a pool of complementary technologies such as sensors, wireless sensor networks (WSNs), radio frequency identification (RFID), semantic technologies, cloud computing, edge computing, big data, AI, blockchains, cybersecurity, and augmented reality (AR). These technologies (e.g., cloud computing and big data) are the digital enablers of the IoT paradigm. Each of them plays a role in the development, deployment, and operation of IoT systems and applications. The rest of this book illustrates many of these technologies and the ways in which they are affiliated with IoT, but also the ways in which they are used in IoT applications.

Given that IoT deployments are not based on a single technology but rather on the integration of multiple technologies, nontrivial IoT systems comprise various components such as:

- *Sensors and actuators:* These provide the means for acquiring information from the physical environment (i.e., sensors) and providing actionable commands to devices and physical processes (actuators). Sensors and actuators enable information technology (IT) systems to interact with the physical world (i.e., the field).
- *Communication components:* These enable the interactions between the distributed nodes of an IoT system.
- *Middleware platforms:* These provide the digital plumbing between devices (such as sensors) and the end-user applications.
- *Data analytics engines:* These undertake the processing of IoT data towards converting raw IoT information to knowledge.
- *Applications:* These provide the user interface (UI) that enables end users to interact with IoT applications.

This list of components is nonexhaustive. In subsequent chapters, we will provide an anatomy of IoT applications through an in-depth presentation of some of these components.

For example, WSN is a special element of IoT systems, which comprises a set of sensors that are configured to form a network. This network of sensors ends up connected to a sink node, which conveys information from the network to other internet-connected entities and devices. WSNs can be considered a component of IoT systems, while at the same time being themselves a special class of IoT applications. WSN was one of the earliest instances of IoT deployments and is, in several cases, considered a forerunner of IoT. Chapter 2 is devoted to the presentation of WSN technologies.

As another example, IoT is very closely affiliated to cloud computing technologies. Sophisticated IoT systems can greatly benefit from the scalability, capacity, and pay-as-you-go characteristics of the cloud. As modern IoT systems deal with large amounts and data, cloud computing provides a compelling value proposition for hosting IoT data and executing IoT services. Moreover, the cloud enables companies to gradually scale up their deployments. As a result, IoT is, in several cases, studied and considered in conjunction with cloud computing. Likewise, most IoT architectures make provisions for the integration of IoT with the cloud. Specifically, the cloud acts as an intermediary between the internet-connected devices and the end-user applications in the various IoT applications areas. It offers services such as storage, computation, and visualization based on popular cloud computing models such as Software-as-a-Service (SaaS), Infrastructure-as-a-Service (IaaS), and Platform-as-a-Service (PaaS). At the same time, there are new cloud computing models for IoT. Chapter 4 is dedicated to the discussion of IoT and cloud integration, where such novel IoT-based cloud computing models are presented.

1.4.2 IoT Applications

There is no single IoT application type but rather a host of different areas where IoT can be applied [8, 9]. Prominent examples include IoT's applications in:

- *Healthcare:* Here IoT opens, for example, new horizons in disease management through a wide range of devices that continually acquire and process information about the context of the patient.
- *Manufacturing:* Here IoT enables new efficiencies both in factory automation and in the efficiency of the supply chain. Specifically, IoT facilitates the seamless flow of information about products and materials across all stakeholders of the supply chain and their information systems.
- *Transport:* Here IoT is enabling completely new modes of transport such as connected cars and self-driving cars, which will radically improve the safety and economic efficiency of driving.
- *Buildings:* Here IoT-based multisensor systems provide the means for efficient management of the buildings, including, for example, advanced features for energy management and facility management.
- *Smart cities:* These provide the urban development context where all the above listed applications and innovations are executed. IoT-enabled smart cities provide advanced functionalities that respond to the challenges of urbanization through alleviating resource depletion and overall improving quality of life. To this end, they integrate a wide range of other IoT-enabled applications such as applications for aging well and efficient urban mobility.

As shown in Figure 1.1, different IoT applications can be developed over a pool of IoT infrastructures that comprise networks, sensors, connected devices, and data. To this end, application developers leverage middleware services provided by IoT infrastructures and middleware.

IoT is completely transforming some of these areas through enabling new disruptive products and services that significantly enhance contemporary service offerings. Chapter 10 presents IoT-enabled applications in the above areas. However,

Figure 1.1 IoT platforms enable the development of different types of IoT applications over networks, sensing, and data infrastructures.

this list of IoT applications is not exhaustive, as the scope of IoT deployments is expanding every day to more application areas.

In some cases, IoT applications are also classified based on their spatial and geographical scope. For example, there are applications for:

- *The home environment (smart home applications):* Such as applications that manage in-home utilities and resources;
- *Communities (smart communities):* Such as applications for renewables and smart energy in urban environments;
- *National scale deployments:* Such as natural resources optimization applications;
- *Transport and mobility applications:* These provide support to users while they are on the move.

These applications are connected to each other through IoT infrastructures, including networks and cloud computing infrastructures. In general, there are many different classifications of IoT applications based on different criteria such as the type of the application, its geographical scope (see Figure 1.2), and its timescale of execution (e.g., real-time or non-real-time).

1.5 Reference Models and Architectures for IoT Systems

In order to understand the elements of an IoT system, one can study reference architecture models, which identify the components that typically comprise an IoT system. Some of the most prominent reference models for IoT systems are presented next.

1.5.1 Industrial Internet Consortium Reference Architecture (IIRA)

The IIRA specifies a common architecture framework for developing interoperable IoT systems for different vertical industries. It is an open, standards-based architecture, which has broad applicability. The latter makes it a vehicle for interoperability, mapping, and practical deployment of IoT technologies, as well as standards

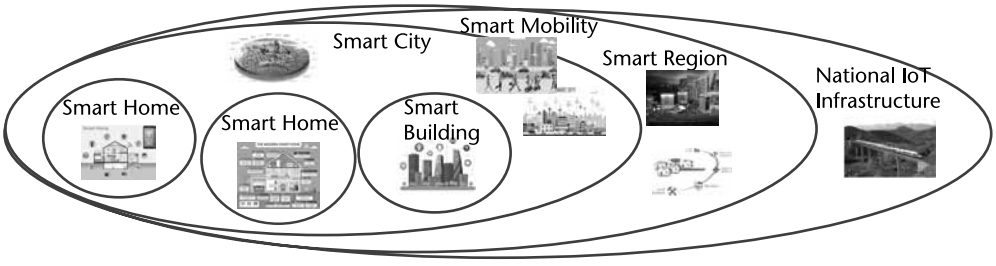

Figure 1.2 The different geographical scope of IoT applications.

development. As a result of its broad applicability, the IIRA is general, abstract, and high-level. Hence, it can be used to drive the structuring principles of an IoT system, without specifying its low-level implementation details. It is also a very good vehicle for communicating concepts and facilitating stakeholders' collaboration.

Based on the analysis of multiple use cases in different sectors, the IIRA presents the structure of IoT systems from four viewpoints, namely, business, usage, functional, and implementation viewpoints. Among these four viewpoints, it is the functional viewpoint that specifies the functionalities of an IIoT system. To this end, the functional viewpoint specifies distinct functionalities in the form of the functional domains. Functional domains can be used to decompose an IoT system in a set of important building blocks, which are applicable across different vertical domains and applications. As such, functional domains are used to conceptualize concrete functional architectures. The IIRA decomposes a typical IoT/Industrial Internet of Things (IIoT) system into five functional domains, namely a control domain, an operations domain, an information domain, an application domain, and a business domain. The implementation viewpoint of the IIRA is based on a three-tier architecture, which follows the edge or cloud computing paradigm. It includes an edge, a platform, and an enterprise tier, which are among the most common elements of a nontrivial IoT system.

1.5.2 OpenFog Reference Architecture

The OpenFog Consortium was a consortium of high-tech industrial enterprises companies and research or academic institutions, which are collaborating towards standardizing and promoting the fog computing paradigm. Fog computing is directly associated with IoT, as it leverages fog nodes (i.e., IoT devices and gateways) in order to enable reliable, low-latency IoT applications. Fog computing alleviates the limitations and drawbacks of conventional cloud computing in various scenarios where low-latency and processing close to the field is required. The reference architecture (RA) of the OpenFog Consortium illustrates the structure of fog computing systems. It presents how fog nodes can be connected partially or fully to enhance the intelligence and operation of an IoT system. Moreover, it presents solutions about growing system wide intelligence away from low-level processing of raw data. The RA is described in terms of different views, including functional and deployment views. The RA specifies some cross-cutting functionalities, which are applied across all layers of an IoT system. These cross-cutting functionalities are conveniently called perspectives.

In January 2019, the Industrial Internet Consortium (IIC) and the OpenFog Consortium (OpenFog) announced that they have finalized the details to combine the two largest and most influential international consortia in industrial IoT: fog and edge computing. Hence, the OpenFog RA is no longer evolving independently. Rather relevant work is conducted in conjunction with the Industrial Internet Consortium.

1.5.3 IoT Reference Architecture Model

A reference model for IoT systems was also developed in the scope of the EU IoT-A project (www.iot-a.eu). The model is currently maintained by the IoT Forum (https://iotforum.org/). IoT-A produced an architectural reference model (ARM) for IoT systems, which addressed interoperability issues that affect nontrivial IoT systems that comprise many heterogeneous components. The IoT-A ARM relies on five different views of submodels.

Specifically, IoT-A comprises a domain model, which defines a set of high-level components or actors that are common to most IoT use cases. Likewise, an information model delves into the analysis, specifying common attributes for each one of the entities of the domain model and their semantics. From the domain model, the IoT-A ARM introduces a set of functionality groups (FGs) as part of the functional model. Specific FGs are included to handle the peculiar characteristics of IoT communication flows. The communication model relies on these components of the architecture and on ISO OSI reference model [10] to address interoperability at the stack or network level. Finally, security aspects are in the scope of a trust, security, and privacy model.

1.5.4 ISO/IEC CD 30141 Internet of Things Reference Architecture (IoT RA)

ISO/IEC CD 30141 Internet of Things Reference Architecture (IoT RA) is an ongoing standardization process. ISO/IEC 30141 performed a heuristic analysis of system characteristics that are common in most IoT systems (e.g., auto-configuration, discoverability, scalability). From them, it proposes a conceptual model where typical IoT entities or actors and their relationships are described. The CM is quite like the IoT-A ARM domain model and includes IoT users, applications, services, or virtual entities. Finally, the five different views that describe the architecture are presented: functional, system, communications, information, and usage.

Other reference models have been produced by vendors, such as Microsoft (e.g., the Azure IoT Reference Architecture [11]) and Amazon [12].

Table 1.1 illustrates provides an overview of the different RAs for IoT Systems. For every RA, it lists the standards development organization (SDO) in charge, the main focus of the architecture, and the target application sectors.

1.6 The Different Layers and Viewpoints of IoT Systems

1.6.1 Layers of IoT Systems

Most of the IoT reference architectures provide layered views of IoT systems [13]. These views provide another good way to understand the various components that comprise and IoT application. While there is no common layered view across the different architecture models, the following layers are in generally found in IoT systems.

Table 1.1 Overview of Different RAs for IoT Systems

Short Name_ Organization in Charge	*Type/Emphasis*	*Target Sector*	
IIRA	Industrial Internet Consortium	Framework for interoperable IoT systems	Various vertical industrial sectors (e.g., manufacturing, energy, healthcare)
OpenFog RA	OpenFog Consortium	Structuring principles for fog computing systems	Smart cities and sectors where fog computing is used
IoT-A ARM	IoT Forum	Reference model for creating IoT architectures	Applicable to all sectors
ISO/IEC CD 30141 IoT RA	ISO/IEC	Conceptual models for IoT systems, actors, and their relationships	Applicable to all sectors

1.6.1.1 Sensing and Actuating Layer

This is the layer where various internet-connected devices reside. These include the various sensors and actuators, but also the servers and gateways to which they are typically attached. The term "sensing" is quite limiting in this context, as IoT enables a much broader range of interactions with the physical world, such as intelligent braking systems or socially assistive robots.

1.6.1.2 Network Layer

This layer ensures the connectivity of the internet-connected objects, based on the various communications and networking technologies. Internet-connected objects connect to IoT systems based on different networks, depending on their appropriateness but mainly their availability. For example, several devices (such as smart phones and wearables) connect to the internet through Wi-Fi hotspots. In practice, Wi-Fi is one of the networks that carry most of the IoT traffic worldwide. Cellular phones connect to the internet through mobile networks of various generations, including 3G and 4G/Long Term Evolution (LTE) networks. There are also cases of objects and devices (e.g., devices within ships), which connect to the internet and IoT based on satellite networks. Based on these networking technologies, IoT deployments leverage functionalities such as capacity management, abstraction of resources, and control of operations. Moreover, most of these technologies are based on Internet Protocol (IP)-based backbone networks.

1.6.1.3 Middleware Layer

This layer converts the low-level networked data source to high-level information consumed by IoT applications towards improving business processes and enabling optimized decisions. Typically, this layer includes middleware platforms that process large amounts of data from the lower layers. The processing involves data

filtering and analytics, including functions for mining large datasets such as big data analytics and AI.

1.6.1.4 Application Layer

The applications layer includes the various applications that can access information and services of the internet-connected devices, through the middleware layer. It also involves the proper visualization of data and services based on dashboards, charts, and more advanced data cyber-representations of data such as virtual reality (VR) and augmented reality (AR).

Figure 1.3 illustrates a layered view of IoT systems based on the above-listed layers.

1.6.2 IoT System Perspectives

Many different stakeholders and actors are involved in IoT systems development and deployment. These stakeholders tend to develop different views and perspectives about IoT. For example, some of them focus on the development of sensors and sensing technologies, including intelligent sensors and energy efficient devices. These people have a "things"-oriented view of the IoT, as they focus on IoT devices, such as smart sensors, smart actuators, and robotic devices.

Other actors consider IoT as a pool of internet-based technologies for accessing information and services on devices. They approach IoT from the perspective of the internet and the web and therefore develop an internet-oriented view of IoT.

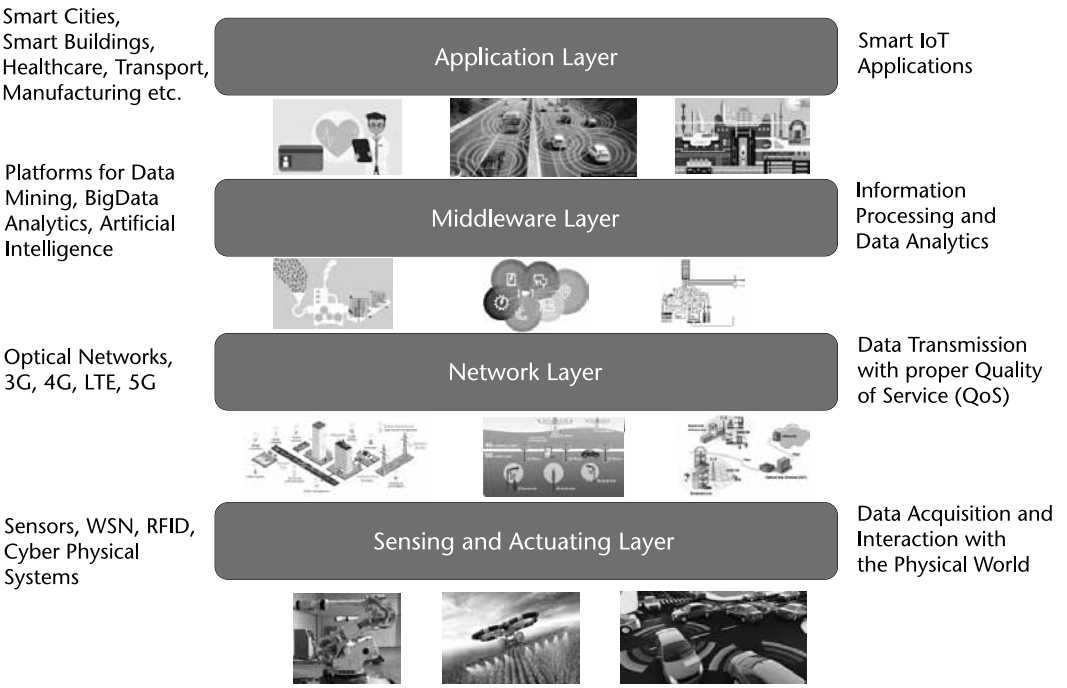

Figure 1.3 Layered structure for IoT systems.

There are also people that work on the semantic unification of the highly heterogeneous landscape of internet-connected objects. They have the semantic-oriented perspective of the IoT.

Other visions and perspectives are also developed. For example, there are actors that perceive IoT as a collection of cloud APIs for application development. Moreover, there are engineers that are completely focused on the networking aspects of IoT. For example, they focus on high-speed connectivity and optimized network quality of service (QoS) for IoT applications.

In practice, IoT is the intersection of all these technologies and visions, especially when it comes to large-scale IoT systems and their integration. Nevertheless, the breadth and depth of IoT technologies make it reasonable for different people to focus on their different areas of expertise. In this book, we try to present many different IoT technologies and applications to provide a 360° view of what IoT is all about.

1.7 Evolution of IoT Systems

For over a decade, IoT systems and applications have been evolving in rapid pace. Along with the proliferation of internet-connected devices, we have also witnessed radical changes in the architectures of IoT systems, as well as in the sophistication of their functionalities. In general, we can distinguish between four phases in the evolution of IoT systems, including (Figure 1.4):

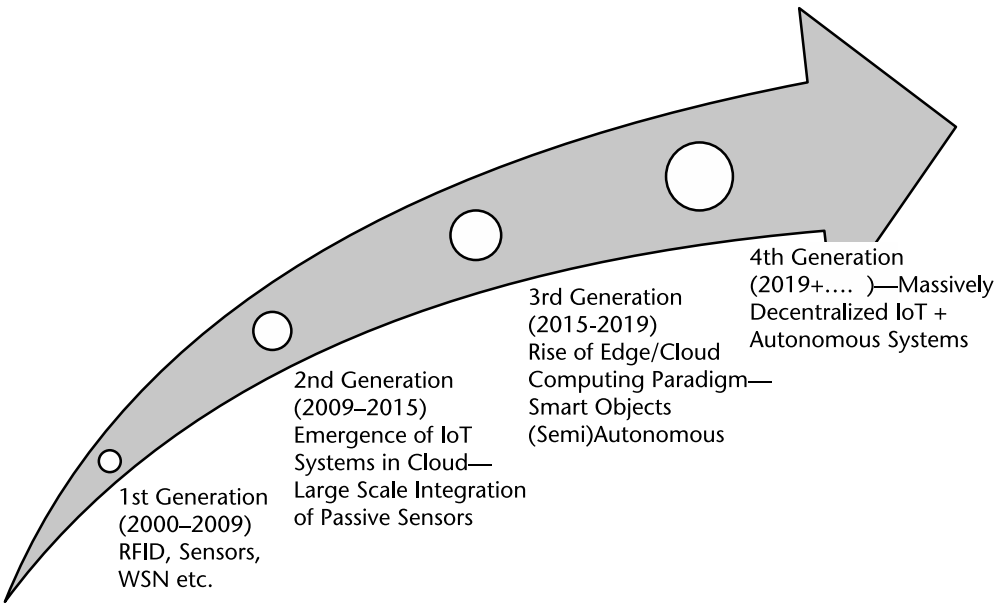

Figure 1.4 Evolution of IoT systems.

- *First generation (2000–2009):* This phase included the development of IoT systems with sensing functionalities based on technologies such as WSN and RFID. The main characteristics of these systems were that they were passive and mainly focused on the collection, analysis, and visualization of data in a pervasive way.
- *Second generation (2010–2015):* As part of this phase, early passive and nonscalable IoT deployments were replaced by cloud-based IoT applications, which provided scalability and elasticity in managing large numbers of passive devices (e.g., sensors). This second generation of the IoT system was driven by the emergence and rise of cloud computing.
- *Third generation (2015–2019):* In this phase, cloud-based IoT systems were gradually decentralized based on the introduction of the edge computing paradigm, which facilitates operations close to the field and offers various performance, scalability, security, and privacy advantages. The notion of computing at the edge of the network had been around several years before. However, it was only in recent years where it was formalized, standardized, and deployed at scale. Note the mainstream reference architectures for IoT systems (such as the IIRA and the OpenFog RA), structure IoT implementations as cloud and edge computing systems.
- *Fourth generation (2019–present):* In recent years, IoT systems are increasingly comprising smart objects with semiautonomous behaviors, such as autonomous guided vehicles, drones, and smart wearables. These objects operate independently of a single edge or cloud administrative domain and take advantage of recently developed edge computing architectures that support mobility and multiple access points such as multi-access edge computing (MEC). The latter enables a higher decentralization of IoT applications. In the coming years, IoT systems may be transformed towards exhibiting a higher degree of decentralization, beyond a single edge or cloud computing infrastructure. In this direction, decentralization technologies such as blockchains are likely to play a role.

In this book, we present the systems and technologies that comprise the first three generations of IoT systems, while developing a vision for future IoT systems. The latter will be driven by the following trends:

- *Proliferation of semiautonomous and mobile smart objects:* In the coming years, the number of smart, autonomous, and mobile objects will explode. For example, recent projections state that by 2030, there will be 700 million connected cars, 90 million (semi)autonomous vehicles, and 250 million electric and hybrid vehicles on the roads, along with 600 million subscribers to various mobility services.
- *Expanded use of AI:* IoT systems are providing a wealth of data, which propels the deployment of AI systems and applications. AI systems are increasingly deployed and used along with IoT systems in areas like agriculture, healthcare, and industry. AI represents one of the fastest growing markets.

According to Tractica, the global AI software market is expected to grow from approximately $9.5 billion in 2018 to $118.6 billion by 2025 [14].

- *Coexistence and collaboration between human and IoT systems:* Despite the rise of robots and automation, humans remain the most flexible and intelligent resource in most IoT applications. In the coming decades, humans are set to coexist and collaborate with IoT systems. Typical scenarios include human-robot collaboration (Cobots) in industrial plants, as well as coexistence of autonomous cars and legacy vehicles in future transport scenarios.

Following chapters explore these trends and how they are likely to impact the future IoT architectures and applications.

1.8 Getting Started with IoT Devices: Sensors and Actuators

The starting point for developers, deployers, and researchers that would like to understand IoT in practice is always sensors and actuators. These are the most fundamental components of an IoT system. They were the main parts of early IoT systems, but they are still an integral element of more sophisticated edge or cloud IoT systems. Hence, they are worth exploring as part of our introduction to IoT systems.

1.8.1 Introduction to Sensors and Actuators

IoT is largely about leveraging data and services from internet-connected devices. Sensors are the most popular IoT devices. A sensor provides usable output in response to a specified measurement. Specifically, it acquires a physical parameter and converts it into a signal suitable for processing (e.g., optical, electrical, mechanical). Sensors are deployed in many different contexts including human bodies, automobiles, airplanes, cellular telephones, radios, chemical plants, and industrial plants. Subsequently, these are deployed and used in many different applications such as healthcare, buildings, transport, energy, water management, and smart cities' applications.

In order to understand the different types of sensors, we present in Table 1.2 different types of stimuli, which are as part of a sensing functions converted to different signals. In particular:

- An acoustic stimulus can produce a wave (amplitude, phase, polarization), spectrum, or velocity related signal.
- A biological and chemical stimulus produces fluid concentrations (i.e., based on gas or liquid).
- An electric stimulus can result in charge, voltage, current, electric Field (amplitude, phase, polarization), conductivity or permittivity signal.
- A magnetic stimulus produces a magnetic field (amplitude, phase, polarization) or flux or permeability signal.

Table 1.2 Sensing Functions

Stimulus Type	*Signal/Quantity*
Acoustic	Wave (amplitude, phase, polarization), spectrum wave, velocity
Biological and chemical	Fluid concentrations (gas or liquid)
Electric	Charge, voltage, current, electric field (amplitude, phase, polarization), conductivity, permittivity
Magnetic	Magnetic field (amplitude, phase, polarization), flux, permeability
Optical	Retractive index, reflectivity, absorption
Thermal	Temperature, flux, specific heat, thermal conductivity
Mechanical	Position, velocity, acceleration, force, strain, stress, pressure, torque

- An optical stimulus can give a refractive index, reflectivity, or absorption signal.
- A thermal stimulus can result in a temperature, flux, specific heat, or thermal conductivity signal.
- A mechanical stimulus can give a position, velocity, acceleration, force, strain, stress, pressure, and torque signal.

Sensors and actuators can both be characterized as transducers. The role of a transducer is to convert a primary form of energy into a signal with a different energy form. The term "energy form" refers to mechanical, thermal, electromagnetic, optical, chemical, and other forms of energy. Specifically:

- A sensor acquires information from the real world, by detecting and measuring a signal or stimulus. We have already presented different stimuli that are commonly used in IoT applications.
- An actuator generates a signal or stimulus, which it usually provides from the digital world to the real (physical) world.

Both sensors and actuators can be deployed as part of an intelligent feedback system to close its loop of operations: The sensor derives information from the real world and converts it to a digital representation of physical quantities. Likewise, the actuator provides to the physical world (for example, a device or physical process) information or a stimulus or trigger produced in the digital world. This closed loop operation is usually an essential part of a CPS.

Based on the general principles of the sensing paradigm, a wide range of sensors have been developed and are commercially available. Furthermore, they are extensively and increasingly used in several applications. Some prominent examples of sensors and their applications are:

- *Temperature sensors:* Temperature can be measured through pressure, volume, electrical resistance, and strain. Temperature sensing is used in several applications and contexts, including buildings, chemical process plants, engines, appliances, and computers.

- *Accelerometers:* These sensors measure along one axis and are insensitive to orthogonal directions. Accelerometer sensors are commonly integrated in most state-of-the-art smart phones. Beyond integration in smart phones, their applications include vibrations, blasts, impacts, shock waves, air bags, washing machines, heart monitors, and car alarms.
- *Light sensors:* They are composed of a photoconductor, such as a photoresistor, photodiode, or phototransistor. Their applications include cameras, infrared detectors, and ambient lighting applications.
- *Ultrasonic sensors:* They are used for position measurements. In the scope of their operation, sound waves emitted are in the range of 2 to 13 MHz. Two typical examples of ultrasonic sensors include Sound Navigation And Ranging (SONAR) and Radio Detection And Ranging (RADAR) sensors.
- *Photogate sensors:* They are typically used in counting applications (e.g., motion period identification). As part of their operation, they record the time(s) at which light is broken. Photogate sensors include infrared transmitter and receiver at opposite ends of the sensor.
- *CO_2 gas sensors:* They measure gaseous CO_2 levels in an environment. Their measurements are in the range of 0 to 5,000 ppm (parts per million) and are based on the monitoring of infrared radiation absorbed by CO_2 molecules.

1.8.2 Sensor Selection Criteria

The list of presented sensors is obviously nonexhaustive. There are tens of different types of sensors, while new types of innovative sensors are continually emerging. A significant part of IoT innovation lies in the sensing part of IoT systems. Novel sensing techniques and mechanisms are in several cases enabling innovative and disruptive applications. There are numerous examples of novel sensors, such as temperature-sensitive RFID tags (i.e., sensor tags) or ingestible pills that ensure adherence to medication schedules. Depending on the target applications, developers and deployers of IoT applications are offered with many different sensor selection options. However, even when deciding about the type of sensor needed, developers and deployers have many different choices in terms of vendors and qualities. In general, the sensors' selection criteria can be classified in three different categories (i.e., cost-related, environment-related, sensor characteristics-related) as follows:

- *Economic criteria:* Economic criteria include the cost, the availability, and the lifetime of the sensor. These cost-related parameters are part of the technical specifications of a sensor. They can be used (along with the cost of purchase of the sensor) to calculate and appropriately plan the total cost of ownership (TCO) of the sensor. For the TCO calculation, it is not only the absolute cost that matters, but also what will be the duration of use and overall availability of the application based on the sensor.
- *Environmental criteria:* Environmental criteria include the size, the power consumption, and the interference and sensitivity of the sensor. This set of criteria is used with a dual objective:

  - To plan and ensure the energy efficiency of the application, including its sustainability;
  - To consider the appropriateness of the sensor for the environment at hand, as the sensing capability and accuracy of a sensor vary based on environmental conditions including, for example, the presence of other sensors or electromagnetic devices.
- *Sensor characteristics:* These include the properties of the sensor, including its sensitivity, range, stability, error, and response time. These characteristics should be evaluated against the requirements of the IoT deployment, which is closely related to the business requirements of the target IoT application. These characteristics should be considered to ensure the feasibility of the target deployment. A sensor that does not come with the proper properties and technical characteristics may not be able to support a given application. Beyond feasibility, the various properties should be audited against their ability to optimize the IoT application from a techno-economic perspective (i.e., ensuring both its technical robustness and quality, but also its value for money).

1.8.3 Simple IoT Devices: Arduino Boards and Raspberry Pi

1.8.3.1 Arduino Boards

In most cases IoT devices host multiple sensors and actuators. Consider, for example, the Arduino board [15], which is one of the most popular IoT devices. Arduino is an open-source electronic prototyping platform, which comprises flexible and easy-to-use hardware and software. It is very popular among several groups of IoT applications, including artists, designers, and hobbyists. In the IoT world, it is also extensively used for IoT education and training, as it provides a low-cost and versatile device for explaining what IoT is about. Arduino is appropriate for applications involving interactive objects or environments.

In practice, Arduino is a microcontroller, which can be deployed as a bridge between the cyber and physical worlds. It balances functionality and ease of use, while it can be acquired at a very low cost. Its price starts from as low as $35. Among the main merits of Arduino is that it is a programmable device through the Arduino C language. The Arduino board comes in various types with different characteristics including Leonard, Due, Micro, LilyPad, Esplora, and Uno.

Among the main reasons for Arduino's popularity is its programmability. Arduino provides the means for programming its operation based on a C++ based language. Arduino programs are called sketches. Each sketch needs at least two functions:

- *void setup ( ):* This runs first and is run once. It is the place where several initializations take place.
- *void loop ( ):* This runs over and over, until power is lost or a new sketch is loaded. It hosts the application logic of the board's operation.

Sketches can access the PINs (Pressure Ins) of the board, as PINs are accessible as global variables. Hence, other Arduino programming functions include:

- *digitalWrite(pin, value):* This sends a voltage level (specified by value) to the designated pin.
- *pinMode(pin, mode):* This designates the specified pin for input or output (as specified by mode).
- *digitalRead(pin):* This reads the current voltage level from the designated pin.

Furthermore, there are also analog versions of above, with the analog read range being between 0 and 1,023. Also, Arduino supports serial commands such as print, println (i.e., print line), and write. Developers can access information and support for programming Arduino through the online support forum at www.arduino.cc.

Another factor that provides flexibility and advantages to the Arduino IoT device is its ability to integrate sensors and shields. Specifically:

- *Arduino sensors:* They can be both binary or of a range. They measure a range of values, through varying their resistance to reflect their detection. Arduinos sense voltages rather than resistances. Moreover, there are sensors that only vary their resistances based on a voltage divider that provides Arduino a voltage as required.
- *Arduino Shields:* Shields are circuit boards that plug into the top of an Arduino to enhance its functionality. Through shields an Arduino can be enhanced with Ethernet modules, Global Positioning System (GPS) modules, motor modules, prototypes, and more.

By enhancing Arduino with sensors and shields, one can build amazing applications based on reasonable effort and at a rather low cost.

1.8.3.2 Raspberry Pi

Raspberry Pi [16] is another popular device that is used for IoT application development in the scope of both education and enterprise applications. It has been developed by the University of Cambridge. Raspberry Pi is a credit card-sized personal computer (PC), which can be plugged into a TV or monitor as a display device. Apart from IoT education purposes, it is used for programming projects, electronic projects, office applications, high-definition (HD) video playback applications, and more. It costs approximately $35 to $100 depending on its extras and the purchased flavor.

Raspberry Pi components include:

- Core components of the PC, including the Raspberry Pi board, a prepared operating system SD card, a universal serial bus (USB) keyboard, a display

(comprising high-definition multimedia interface (HDMI), digital visual interface (DVI), or composite input), and a power supply;

- Extras (i.e., optional components), including USB mouse, a local area network (LAN) cable for Internet connectivity, a powered USB hub, and a case;
- Programming languages, which enable the implementation of applications over the PC. Supported languages and associated environments include C, C++, Java, Scratch, and Ruby, but also any language supported on the ARM6 processor. These languages provide great flexibility in the implementation of the applications.

Like Arduino, the Raspberry board provides opportunities for implementing fantastic applications, both for enterprise use and for fun projects. Three simple applications that illustrate the concept of extending Raspberry Pi with IoT-based functionalities are:

- *A simple programmable networked sensor:* This is a low-cost digital temperature and humidity sensor integrated with Raspberry Pi. It is an open-source project available at GitHub at: https://github.com/jervine/rpi-temp-humid-monitor.
- *ThinkBox (IoT on Raspberry Pi):* This is software already installed and configured on Raspberry to enhance it with IoT development capabilities through a visual UI. It enables graphical creation of new applications from a simple web browser. More information about the ThinkBox project is available at http://thethingbox.io/.
- *Raspberry People Counter:* This application is based on motion sensors that are wired to the GPIO pins of Raspberry Pi. This is the development of a people counting application [17].

1.9 Summary

This chapter presented an overview of the IoT computing paradigm and of its main components and technologies. It also described some of the popular reference architectures for IoT systems, which provide the means for integrating IoT technologies in sophisticated applications. This chapter described in more detail some of the basic components of IoT applications such as sensors and Arduino boards. These components can be used for developing IoT simple applications for educational purposes. Real-life IoT applications comprise more sophisticated IoT technologies. The rest of the book dives into the details of more advanced IoT components and technologies starting from WSN. WSNs comprise multiple networked sensors and are the natural next step in our IoT journey, which is continued in Chapter 2.

References

[1] International Telecommunication Union, "Y.2060: Overview of the Internet of Things," June 15, 2012, https://www.itu.int/rec/T-REC-Y.2060-201206-I.

[2] Atzori, L., A. Iera, and G. Morabito, "The Internet of Things: A Survey," *Computer Networks*, Vol. 54, No. 15, 2010, pp. 2787–2805.

[3] Help Net Security, "Number of Active IoT Devices Expected to Reach 24.1 Billion in 2030," May 22, 2020, https://www.helpnetsecurity.com/2020/05/22/active-iot-devices/.

[4] Zorzi, M., et al., "From Today's Intranet of Things to a Future Internet of Things: A Wireless- and Mobility-Related Views," *IEEE Wireless Communications*, Vol. 17, No. 6, 2010, pp. 44–51.

[5] Soldatos, J., O. Lazaro, and F. Cavadini, (eds.), *The Digital Shopfloor: Industrial Automation in the Industry 4.0 Era*, Denmark: River Publishers, 2019.

[6] Boussard, M., et al., "Providing User Support in Web-of-Things Enabled Smart Spaces," *Proc. of 2nd International Workshop on Web of Things*, Vol. 11, June 2011.

[7] Guinard, D., et al., "From the Internet of Things to the Web of Things: Resource Oriented Architecture and Best Practices," in *Architecting the Internet of Things*, New York: Springer, 2011, pp. 97–129.

[8] Perwej, Y., et al., "An Extended Review on Internet of Things (IoT) and Its Promising Applications," *Communications on Applied Electronics (CAE)*, Foundation of Computer Science FCS, New York, Vol. 9, No. 26, February 2019, pp. 8–22.

[9] Da Xu, L., W. He, and S. Li, "Internet of Things in Industries: A Survey," *IEEE Transactions on Industrial Informatics*, Vol. 10, No. 4, November 2014, pp. 2233–2243.

[10] International Standardization for Organization, "ISO 7498-1 – Open Systems Interconnection," November 1994, http://standards.iso.org/ittf/PubliclyAvailableStandards/s020269_ISO_IEC_7498-1_1994(E).zip.

[11] Microsoft, "Azure IoT Reference Architecture," January 9, 2019, https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/iot/.

[12] AWS, "Reference Architecture," https://aws.amazon.com/lambda/resources/reference-architectures/.

[13] Gubbi, J., et al., "Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions," *The Computing Research Repository (CoRR)*, July 2012.

[14] OMDIA/Tractica, "Artificial Intelligence Software Market to Reach $118.6 Billion in Annual Worldwide Revenue by 2025," April 30, 2019, https://tractica.omdia.com/newsroom/press-releases/artificial-intelligence-software-market-to-reach-118-6-billion-in-annual-worldwide-revenue-by-2025/.

[15] Arduino, 2020, https://www.arduino.cc.

[16] Raspberry Pi, 2020, https://www.raspberrypi.org/.

[17] Pelaez, A., "Building a People Counter with Raspberry Pi and Ubidots," August 23, 2013, http://blog.ubidots.com/building-a-people-counter-with-raspberry-pi-and-ubidots.

CHAPTER 2

Wireless Sensor Networks

This chapter presents wireless sensor networks (WSNs), one of IoT's forerunner technologies, which is still one of the main elements of several IoT deployments. Specifically, the chapter presents WSN technologies across three main axes:

- First, it introduces the WSN concept, along with the main components of a WSN system.
- Second, it discusses WSN middleware as an essential element of a WSN deployment. WSN middleware provides WSNs with some of their most important features such as energy efficiency and the ability to interface to heterogeneous sensors and devices.
- Lastly, it presents some examples of WSN applications and deployments, along with some examples of WSN middleware.

2.1 Introduction to WSN

The motivation behind WSN is that the largest portion of computing processes resides in devices other than traditional desktop computing systems. Sensing capabilities are deployed within household appliances, vehicles, industrial machinery, and many other forms of industrial and consumer devices. This creates opportunities for new added-value infrastructures and applications based on the ubiquitous networking of the billions of physically embedded computing devices. The latter include numerous sensing devices. Such infrastructures and applications can be developed based on the integration of reliable wireless communication and sensing functions to such physical devices (i.e., "things"). In this context, we define a distributed WSN as a collection of embedded sensor devices with networking capabilities. WSNs are also characterized as deeply networked systems or pervasive networks [1, 2].

In the scope of Chapter 1, we already defined and presented the basic operations of a sensor. At this point, we are revisiting the definition and description of a sensing system. Specifically, a sensing system is typically composed of:

1. Some memory;
2. A device enabling communication with other systems;
3. A controller that coordinate interactions between the various modules of the sensing system;

4. An actual sensing or actuator device.

Wireless sensors comprise these components, including an integrated wireless transceiver that enables wireless communications.

Sensing systems are nowadays proliferating as a result of continuous advances in microelectromechanical systems (MEMS) technologies. However, a typically sensing system comes also with limitations in terms of energy, computation, storage, transmission range, and bandwidth.

A WSN consists of multiple sensor nodes. Each sensor node is made up of four basic components, including a sensing unit, a processing unit, a transceiver unit, and a power unit. Specifically:

- *Sensing units:* These are usually composed of two subunits: sensors and analog-to-digital converters (ADCs). The analog signals produced by the sensors are converted to digital signals by the ADC and then fed into the processing unit.
- *Processing unit:* This comes with a small storage unit. It manages the procedures that make the sensor node collaborate with the other nodes to carry out the assigned sensing tasks.
- *Transceiver unit:* This connects the node to the network.
- *Power units:* These are supported by a power scavenging unit, such as solar cells.

Other subunits of the node are application-dependent and are likely to vary depending on the target application and the deployment context of the WSN. For example, application-dependent components may include a location finding system, a power generator, and a mobilizer.

As already outlined, a WSN comprises multiple sensor nodes such as a sensor array. The sensors used in the application are the end points that communicate real-time data to the strategically positioned base stations in the coverage region. The various sensors provide a wide coverage of the target area in line with application requirements. Furthermore, the inputs of the various sensors are aggregated and integrated in the base station as needed by the WSN application. A typical WSN deployment can comprise hundreds to thousands of sensor nodes. The latter collect environmental information and send it toward a sink node. Sink nodes are the delivering information to end users (Figure 2.1).

WSN deployments provide clear advantages over single sensor deployments. These advantages include:

- *Extended sensing ranges:* They can cover a broader area than a single sensor.
- *Redundancy:* They provide increased resilience and fault tolerance. When a sensor node fails, a neighboring node is usually able to provide the needed information.
- *Improved accuracy:* This happens because nodes collaborate and combine their data in order to increase the accuracy of their measurements.

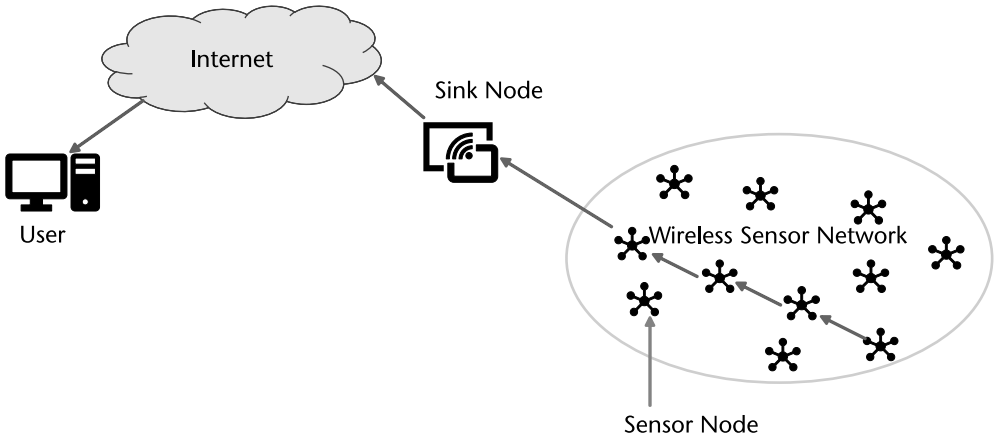

Figure 2.1 WSN overview.

- *Enriched functionalities:* These happen because sensing and forwarding services provided by sensing nodes can be combined in multiple ways in line with the requirements of the target application.

2.2 WSN Applications

WSN applications provide the means for monitoring physical systems and phenomena in various application areas. WSN applications are considered IoT applications as they comprise devices that monitor the physical world. However, state-of-the-art IoT applications comprise not only monitoring functionalities, but also actuation and control functionalities.

WSN applications consist of a data collection network, which is used to collect information from numerous distributed devices or even other WSNs. This information arrives to a base station controller and is then transmitted to consumers of WSN information. The latter are usually terminal devices such as smartphones, personal computers, and servers. In the scope of a WSN system, we can distinguish between the data acquisition network and the data distribution network. The data acquisition network collects information from the various sensors. The data distribution network undertakes the dissemination of the information to the various consumers and client devices. In large-scale deployments, the data distribution network is likely to comprise multiple networking infrastructures such as wireless local area networks (LANs) and cellular networks. Furthermore, large-scale WSN deployments may comprise multiple WSNs spanning various administrative domains and application sectors.

WSNs can support applications in multiple sectors. A nonexhaustive list of WSN application domains and areas include [3]:

- Physical security for military operations;
- Indoor and outdoor environmental monitoring;

- Seismic and structural monitoring;
- Industrial automation;
- Biomedical applications;
- Health and wellness monitoring;
- Inventory location awareness;
- Future consumer applications, including smart homes.

2.3 WSN Characteristics and Design Challenges

Most WSNs share the following main features:

- *Deeply distributed architecture:* WSNs are highly distributed and not subject to central control. Rather, they feature localized coordination as required to meet system and application goals.
- *Autonomous operation:* Several WSNs operate in an autonomous fashion. Specifically, they are usually characterized by self-organization, self-configuration, and adaptation to changing conditions.
- *Energy conservation:* WSNs are designed to be as energy-efficient and independent as possible. To this end, they implement energy optimization features at multiple levels of the network (e.g., at the physical layer and at the media access control (MAC) layer). Furthermore, they tend to deploy energy efficient routing and efficient application interactions [4].
- *Scalability:* WSNs should be scalable both in terms of node density and in terms of the number and type of supported nodes. Scalability is a key prerequisite for supporting large-scale applications.
- *Data centricity:* Several WSNs are data-centric. This means that they operate based on the data that they collect and transfer. For example, the routing of the information is based on the carried data rather than on a network address. Furthermore, data-centric WSNs support functions such filtering and aggregation of the data within the network.

The design of a WSN is usually driven by the requirements of the target application. In principle, the WSN designer sets one or (usually) more of the following goals:

- *Power efficiency:* WSNs must be power efficient, as they tend to operate based on limited battery power. Power efficiency can be achieved in various ways, including the design of energy efficient and computationally efficient protocols, as well as based on the optimization of input-output (I/O) operations of the nodes of the network. Overall, power consumption optimization takes place at different levels and functions, including sensing, communication, and data processing functions.

- *Storage and computational efficiency:* WSNs must be also storage efficient, as sensor nodes have limited storage capacity. Likewise, sensor nodes should be able to operate with limited computation, as their central processing unit (CPU) is limited as well.
- *Bandwidth efficiency and error rate minimization:* WSNs must be bandwidth efficient, as they operate in bandwidth constraint environments. Moreover, they must minimize their error rates. Unless errors are controlled, WSNs will provide data with high veracity, which makes application development challenging.
- *Scalability:* Scalability refers to the density of the sensor nodes. Nontrivial WSNs can comprise thousands of nodes. Hence, WSN designers should support their scalability. In practice, a designer needs to estimate and deploy the number of nodes required within a given radio transmission range. It is also important to ensure that WSN scaling occurs in a cost-effective manner (i.e., without a need for additional investments that grow linearly based on the number of nodes of the WSN).
- *Self-configured operations:* In several cases, a WSN must be self-configurable. This is required in cases where they should operate in an autonomous fashion (i.e., without support from an infrastructure network).
- *Limit memory footprint:* In addition to power efficiency, there is a need for designing and implementing protocols with a limited memory footprint. This is required because sensor nodes have limited memory.
- *Simple and efficient protocols for nodes interaction:* As an extension to the computational and storage efficiency of the network, there is also a need to design and implement simple and efficient protocols between the nodes of the network.

Given these challenges, WSN designers must resolve some trade-offs when designing their network. Specifically, they need to make choices regarding:

- *Fault tolerance:* Fault tolerance challenges refer to the ability of sustaining the sensor network functionalities without interruption due to sensor node failures. This challenge is application-dependent as the desired fault tolerance level should be aligned to the level of resilience required by the application. By relaxing fault tolerance needs, designers can produce power and computational efficient networks.
- *Production costs:* WSN production costs must be kept at acceptable levels. In this direction, the cost of a single node is very important. The cost factor becomes very challenging given the increasing sophistication of WSN nodes. Production costs must be traded off with the levels of power efficiency, computational efficiency, and scalability of the WSN.
- *Sensor network topology:* The deployment of a proper network topology is very important in practice. Deployers are offered with an opportunity to tackle the issue of finding a proper topology in several phases, namely a

pre-deployment phase, a deployment phase, and a post-deployment phase. As part of the latter phase, redeployment of additional nodes is possible. The deployment of additional nodes can fine-tune the topology of the WSN in a way that fulfills requirements like power efficiency, scalability, and fault tolerance.

As part of the practical deployment of a WSN, the following should be also considered:

- *Deployment environment:* This impacts the operation of the WSN in terms of accuracy and data quality. The data accuracy and the overall performance of a network can be radically different when the WSN is deployed at different contexts. For instance, harsh environments make accuracy and performance more challenging. Some examples of harsh environments are busy intersections, the interior of large machinery, the bottom or the surface of an ocean, a biologically or chemically contaminated field, a home environment, a large building, and a warehouse.
- *Transmission media:* This can have a major impact on the operation and performance of the network. Different transmission options (e.g., radio, infrared, optical) lead to networks of varying performance. Moreover, in cases where operations must span very large areas, a transmission medium that is available in these areas must be used.

2.4 WSN Layers

2.4.1 Overview

A WSN is a network and hence its main functions can be illustrated based on a layered architecture following the principles of the popular open systems interconnection (OSI) stack [5]. In particular, the operation of a WSN can be illustrated based on five different layers:

- *Physical layer:* This layer provides simple but robust modulation, transmission, and receiving techniques. It includes frequency selection and carrier frequency generation functionalities, along with signal detection and propagation. Moreover, it comprises the necessary signal modulation and data encryption functionalities.
- *Data link layer:* This layer ensures the multiplexing of data streams, along with data frame detection, medium access, and error control. Key functionalities include MAC, power-saving modes of operation, and error control.
- *Network layer:* This layer implements part of the power efficiency characteristics of a WSN. For data-centric WSN, it also implements data aggregation functionalities. Data aggregation is useful only when it does not hinder the collaborative effort of the sensor nodes. As part of this layer, WSNs implement, for example, attribute-based addressing and location awareness.

- *Transport layer:* This layer is required when the network is to be accessed through the internet or other external networks. Conventional Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) meet most requirements.
- *Application layer:* This layer implements management protocols, which ensure that lower layers are transparent (i.e., hardware and software agnostic) to the sensor network management applications. Some popular management protocols are [6]:
  - The Sensor Management Protocol (SMP);
  - The Task Assignment and Data Advertisement Protocol (TADAP);
  - The Sensor Query and Data Dissemination Protocol (SQDDP).

The WSN layered stack enables sensor nodes to communicate based on open and agreed protocols, much in the same way that OSI systems interact in the scope of other networking infrastructures. Figure 2.2 illustrates these layers of a WSN system. It also shows the part (hardware, firmware) of the sensor node where each layer is implemented in practice.

2.4.2 The Data Link Layer

Like in the OSI model, each layer can be divided into sublayers that define fine-grained functionalities. Hence, the main functionalities of the data link layer of a WSN are represented through the following sublayers:

- *MAC:* This deals with the creation of the network infrastructure and the fair and efficient sharing of communication resources between sensor nodes. WSN MAC protocols are usually based on existing MAC protocols for

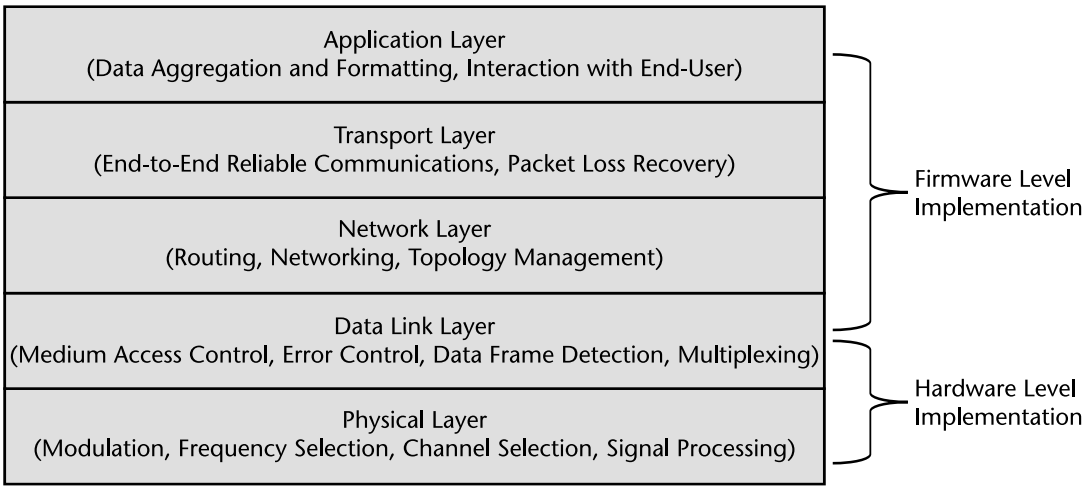

Figure 2.2 Layers of a WSN system.

wireless/mobile systems (e.g., cellular systems, Bluetooth, and mobile ad hoc networks).

- *Error control:* This is implemented in special modes called error control modes. Error control modes in communication networks have additional energy costs for retransmission and include forward error correction (FEC) and automatic repeat query (ARQ) functions, which are typical error control functions in communication networks. In principle, WSNs need simple error control codes with low complexity encoding and decoding.
- *Power-saving modes:* These are sensor nodes communicating using short data packets. Shorter packets lead to start-up energy dominance. In practice, an energy-efficient node operates in a power-saving mode for a period greater than a certain threshold.

2.4.3 The Network Layer

In data-centric WSNs, functions such as data filtering and aggregation take place at the network layer. Specifically, the following functionalities are implemented:

- *Interest dissemination:* Interest dissemination ensures that the sensor nodes will appropriately notify the sink node when specific conditions on the data are met. To this end, the sink broadcasts the interest of the application regarding dissemination of information. Accordingly, sensor nodes broadcast the advertisements. The interest is denoted based on an attribute-based naming approach (e.g., "The areas where humidity is over X").
- *Data aggregation:* Data aggregation is aimed at alleviating implosion and overlap problems. Data is aggregated on the same attribute of a phenomenon in order to consolidate information and save bandwidth and energy. To this end, the locations of reporting sensor nodes are taken into account.

2.4.4 The Application Layer

The Sensor Management Protocol (SMP) is implemented at the application layer to ensure proper management of the WSN resources and to deliver the information required by applications. SMP comprises rules for data aggregation, attribute-based naming, and clustering of the sensor nodes. Its functionalities include:

- Exchanging data needed for the algorithms that utilize the location of the sensor nodes;
- Time synchronization of the sensor nodes to ensure that all nodes use and refer to the same clock;
- Moving sensor nodes, in cases where this is needed by the application;
- Turning sensor nodes on and off, in line with application requirements;
- Querying the sensor network configuration and the status of nodes, in order to make them available to the application;

- Reconfiguring the sensor network in terms of sensor nodes status and location;
- Authentication, key distribution, and security in data communications.

The SQDDP is another application layer protocol for WSNs. It provides interfaces for:

- Issuing queries;
- Responding to queries;
- Collecting incoming replies.

It supports attribute-based naming and querying of the respective nodes, as well as location-aware naming and respective querying. Through SQDDP, it is possible to address nodes either by their attributes or by their location. However, SQDDP does not typically issue queries to individual nodes. It is rather implemented in a way that supports queries addressed to the entire WSN.

2.4.5 Cross-Layer Functionalities

In addition to the layered functionalities, a WSN performs some cross-cutting functions that are layer-agnostic. These functions are implemented based on all the layers of the network. In the scope of computing network such functionalities are usually characterized as planes. For example, the management plane of a network provides management, monitoring, and configuration services to all layers of the network stack and other parts of the system.

In the scope of a WSN, the following three families of cross-cutting functionalities can be defined:

- *Power management:* This deals with power efficiency across all layers of the network.
- *Mobility management:* This aims at supporting the mobility of the nodes in networks that provide mobility.
- *Task management:* This focused on managing the delivery of tasks (e.g., queries) that are assigned to the various nodes of the network as part of its operation.

2.5 WSN Middleware

2.5.1 Middleware Functionalities

Middleware refers to system software that is used for interconnecting systems, protocol conversions, and other system level tasks. In the case of WSNs, middleware plays a significant role in ensuring the networks' scalability, reliability, power efficiency, and other nonfunctional characteristics [7]. Specifically, WSN middleware

is software interconnecting network hardware, operating systems, network stacks, and applications comprising wireless sensor networks. It provides standardized system services to diverse applications, along with a run-time environment for supporting and coordinating multiple applications. Furthermore, it enables adaptive and efficient utilization of system resources.

Earlier in this chapter, we explained the challenges of designing an efficient WSN, along with some options available to designers. At this point we extend the discussion of these challenges to WSN middleware, keeping in mind that WSN middleware is an essential and integral element of every WSN deployment. In particular, the WSN middleware challenges include:

- *Availability of limited power and resources:* Tiny devices are limited in energy and resources such as CPU and memory. As a result, WSN middleware should implement techniques for efficient processing and use of memory, as well as for low-power communications (e.g., sleep modes, intelligent transmission).
- *Scalability, mobility, and dynamic network topology:* WSN middleware provides the means for handling device failures and moving obstacles, especially in cases where mobility and interference result in frequent network changes. Moreover, WSN middleware ensures the high-performing and robust operation of the WSN despite dynamic network changes. Furthermore, at the level of WSN middleware, fault tolerance and sensor node self-configuration and self-maintenance can be implemented.
- *Heterogeneity of resources:* WSNs are highly heterogeneous, as they comprise heterogeneous computing resources and devices, such as CPU power, networking, memory, storage, and operating systems. WSN middleware implementations provide the means for interfacing to numerous hardware, software, and networking resources.
- *Dynamic network organization:* In WSNs, conventional client-server interactions are not always possible. Therefore, in several cases ad hoc discovery of resources and their locations should be implemented considering latency, reliability, and energy trade-offs. Such discovery and location awareness characteristics are typically implemented at the level of the WSN middleware.
- *Real-world integration:* Most WSN applications deal with real-time phenomena, which ask for real-time services. In several cases, WSN applications have to process sensor information in real time. Relevant support can be provided at the level of WSN middleware.
- *Blending of application requirements:* WSN middleware provides the means of matching application-specific requirements to the capabilities of the network. This matching must resolve trade-offs between application specifics and the need for general-purpose middleware.
- *Data aggregation:* Data aggregation is one of the primary mechanisms for alleviating redundant data that increase communications cost. The latter can in several cases surpasses computational cost. WSN middleware aggre-

gates data towards eliminating redundancy and minimizing the number of transmissions to the sink node.

- *Security:* WSN middleware development is driven by security-by-design principles. A WSN must address multiple security requirements including authentication, integrity, data freshness, and availability.

2.5.2 Types of WSN Middleware Platforms

2.5.2.1 Virtual Machines

There are different types of WSN middleware, operating in various ways and serving different purposes. One of the most popular WSN middleware models is that of a virtual machine (VM), which is similar to the VM concept in distributed systems [8]. It provides applications with semantic transparency and independence from the underlying physical sensing infrastructure through the implementation of a common unified software layer. The benefits of a VM are that it provides a common abstraction and a uniform sandbox for all applications. However, the disadvantage of the VM approach is that the virtualized software layer introduces high overhead, while at the same time making it difficult to exploit the various heterogeneous resources. A prominent example of VM middleware for WSN is the TinyOS [9] operating system (OS) [10]. It is an open source, Berkeley Software Distribution (BSD) licensed OS for low-power wireless devices such as sensor networks, personal area networks, and smart meters. TinyOS is very popular with an average of many thousands of downloads per annum.

2.5.2.2 Mobile Agents and Sensor Databases

Other approaches to WSN middleware include mobile agents and sensor databases. Mobile agents enable a modular programming approach. Their benefit is that only parts of the WSN middleware need to be updated, while changes can be propagated efficiently from node to node. However, mobile agents have high overhead and do not allow for the exploitation of heterogeneous hardware. Sensor databases can be used to abstract entire sensor networks as virtual relational databases. This is an approach that eases interoperability with legacy systems, as the latter can also write in the same database. On the downside, sensor databases do not support real-time applications and provide only approximate results.

TinyDB (originally developed at Berkeley) is an example of a sensor database [11]. It is a query processing system for extracting information from a network of TinyOS sensors, which obviates the need for writing embedded C code for sensors. TinyDB provides a simple, SQL-like interface for developers and users to specify the data that they want to extract, along with additional parameters such as rate at which data should be fetched and refreshed. A typical TinyDB application is one that collects data from motes in the environment, filters them, aggregates them, and ultimately routes them out to a host controller. Also, TinyDB implements power-efficient, in-network processing algorithms, which increases the energy efficiency to TinyDB-based applications.

Another example of an WSN middleware framework is the Global Sensor Networks middleware [12], a programmable, open-source middleware framework for sensor networks, which has been developed by EPFL (École polytechnique fédérale de Lausanne) in Switzerland [13]. The development of the middleware has been driven by common requirements for data processing, management, and interfacing (i.e., the middleware is based on the specification of common functionalities in these areas). GSN implements the concept of a virtual sensor (VS), which represents a data stream structure that includes various attributes for the data stream. It is based on a multilayered architecture, which is deployable on resource constrained devices.

GSN is a programmable middleware framework, which can be considered a sensor database as well, given that sensor data are stored in a database. A VS abstraction comprises the following:

- Input and output attributes;
- Various streams;
- Queries over the streams (i.e., the data to be provided by the VS).

This structure of the VS sensor drives the specification of the database schema of the database system where the sensor data are stored. GSN offers a simple SQL-like language for applications development, which supports queries such as the following:

```
Select Image from Camera
Select Temperate from MoteA
Select Camera.Image
from A[window 1],B[window 10min]
where avg(temperature) >30
```

2.5.2.3 Message-Oriented Approaches

WSN middleware systems can be also implemented based on message-oriented approaches. The latter implement publish-subscribe mechanisms that enable asynchronous communications between data producers and consumers within the network. Publish-subscribe communications enable consumers to subscribe for data to one or more sensor nodes. Hence, they ensure that their data will reach the consumers when certain criteria are met. Message Queue Telemetry Transport (MQTT) is a very popular message-oriented approach, which has the advantage of ensuring a loose coupling between the sender and the receiver. MQTT is used in WSNs and also in more complex IoT applications, where its use is expanded for a broader range of data-producing and data-consuming devices. The main disadvantage of the MQTT approach is that messaging introduces overhead to the WSN application.

TinyMQ [14] is an example of an open source implementation of the MQTT approach for WSN. It provides a channel-based, in-memory message queue for handling messages, in line with Erlang mechanisms. Channels are identified by strings and are automatically created and destroyed as needed. Messages are internally

stored in a priority queue (i.e., based on first in-first out scheme). There is also a standards-based variation of TinyMQ, namely, the MQTT for Sensor Networks variation, which is an ISO standard (ISO/IEC PRF 20922).

2.6 WSN Standards

The popularity of WSN applications has given rise to the emergence of various standards. The latter are destined to facilitate the interoperability of different WSNs, as well as the implementation of secure and reliable two-way wireless communications in WSNs. Most of the WSN standards are targeting industrial applications of WSNs in areas such as smart building, oil, gas, and energy sectors. Some of the most prominent WSN standards are described next.

2.6.1 IEEE 802.15.4

IEEE 802.15.4 is a fundamental standard for many WSN technologies, including Zigbee, ISA100.11a, WirelessHART and 6LoWPAN. It defines the physical and media access control layers of low-rate wireless personal area networks (LR-WPANs). Hence, it is focused on lower WSN layers. Building on these layers, the above-listed WSN technologies (e.g., ISA100.11a, WirelessHART) defined higher-level user layers for WSN applications.

IEEE 802.15.4 is destined to support low-cost, low-speed communication between devices. Its focus on WSNs is reflected on its support for low-power, low-bandwidth communications. As such, it is unique when compared to other IEEE standards for wireless networks such as Wi-Fi, which are designed to support bandwidth-savvy applications, yet require more power.

2.6.2 International Society of Automation (ISA) 100 Committee Standards

Several WSN standards have been developed by the ISA100 Committee [15]. The committee has also provided recommended practices, technical reports, and related information that define technologies and procedures for implementing wireless systems. The ISA100 standards and recommendations are primarily focused on the automation and control environment with a focus on the field level (Level 0). Emphasis is put on standards for manufacturing automation applications. The ISA100 standards are destined to provide guidance for designing, implementing, maintaining, and managing control systems that comprise WSN.

The most prominent ISA100 standard for WSN is the ISA100.11a wireless networking technology standard, which is officially described as "Wireless Systems for Industrial Automation: Process Control and Related Applications." Other standards of the ISA 100 family include reference models for WSN management (i.e. the ISA-TR100.20.01-2017 "Common Network Management: Concepts and Terminology"), standards for secure and trusted WSN architectures (i.e. ANSI/ISA-TR100.15.01-2012 "Backhaul Architecture Model: Secured Connectivity over Untrusted or Trusted Networks"), standardized requirements for WSN deployments

in factory automation applications (i.e. ISA-TR100.00.03-2011 "Wireless User Requirements for Factory Automation"), and various comprehensive guides for the development and deployment of wireless systems in applications such as industrial automation and asset tracking.

2.6.3 WirelessHART

WirelessHART is a wireless sensor networking technology that has been developed based on the Highway Addressable Remote Transducer (HART) Protocol. It addresses the requirements of process field device networks and aims at boosting interoperability in multivendor environments.

WirelessHART networks are flat mesh networks that comprise multiple radio stations (i.e., field devices). Each station operates as both a signal source and a repeater. In the scope of a WirelessHART network operation, messages are passed from a transmitter to its nearest neighbor. The process continues until the message reaches the base station and the intended receiver. The technology leverages also alternative routes to alleviate cases where a message cannot be transmitted to given path. To this end, alternative routes are set up in the initialization phase of a WirelessHART network. Overall, WirelessHART networks can extend their range, while providing redundant and thus more resilient communication routes.

2.6.4 ZigBee

ZigBee has been created by the ZigBee Alliance and is a technological standard created for control and sensor networks. It is based on the IEEE 802.15.4 standard. The main targets and properties of ZigBee are:

- Support for low data rate and low power consumption;
- Support for small packet devices.

ZigBee operates in unlicensed bands of the wireless spectrum, specifically:

- The ISM 2.4-GHz global band at 250 kbps;
- The 868-MHz European band at 20 kbps;
- The 915-MHz North American band at 40 kbps.

Moreover, ZigBee operates in personal area networks (PANs) and device-to-device networks. It ensures the connectivity between small packet devices. Typical ZigBee applications can be found in the smart home environment, including control of lights, switches, thermostats, and appliances.

ZigBee functionalities can be represented in a layered model stack (i.e., the ZigBee). The upper layers of the stack (i.e., API, security, network) are specified by the ZigBee Alliance and implemented as WSN software and middleware. However, the lower layers are specified in the IEEE 802.15.4 standard and support the MAC and physical layers of the ZigBee WSN applications.

Table 2.1 provides a short overview of the WSN standards in terms of some of their characteristics.

2.7 Future Trends in WSN

WSN is an established technology that is being commercially deployed in different settings, including IoT systems and applications. However, there are still several areas where novel solutions are researched. Many of these areas concern research that tries to address the WSN challenges presented earlier. Specifically, some of the most prominent research trends for WSN include:

1. *Cross-layer protocols for sensor networks:* The vast majority of WSNs operate based on the conventional layered architectures that were described previously. Nowadays, there is ongoing research in cross-layer protocols, which merge common protocol layer functionalities into cross-layer modules. Such cross-layer modules hold the promise to offer improved resource efficiency, through considering and optimizing transport, routing, and MAC functionalities at the same time. Likewise, cross-layer modules should comprise communication protocols that ensure efficient and reliable communications across sensor nodes.
2. *Energy-efficient protocols for WSN:* The ever-important issue of developing power-efficient protocols for WSN is still in the foreground. Specifically, there is a need for energy-efficient techniques that consider new types of sensors, as well as the rising sophistication of WSN applications.
3. *Energy harvesting and scavenging for WSN:* In the quest for energy-efficient solutions, many researchers are considering energy-harvesting solutions. The latter include the development of specialized sensor nodes that permit capturing and storage of energy from external sources. In the near future, the number of small, wireless, autonomous, energy-scavenging devices will increase to enable novel, power-efficient applications.
4. *Security and privacy architectures for WSN:* Nowadays, there are several WSN applications that collect and process private, sensitive data. For example, this is the case with healthcare applications. Hence, there is a need for researching WSN architectures that provide strong security, privacy, and data protection features, such as encrypted communications, security

Table 2.1 Overview of WSN Standards

Feature	*ISA100.11a*	*WirelessHART*	*ZigBee*
Topology	Mesh, star, hybrid (mesh/star)	Mesh, star, hybrid (mesh/star)	Mesh
Radio channel	Time division multiple access (TDMA)/code division multiple access (CDMA)	TDMA	CSMA-CD
Keys	Symmetric/asymmetric	Symmetric	Symmetric
Deterministic reliability	Yes	Yes	No

monitoring, and authentication functionalities [16]. Likewise, WSNs are increasingly designed based on security-by-design and privacy-by-design features.

5. *High-performance protocols for WSN:* There are still several research initiatives that strive to improve the efficiency of WSN protocols, including transport protocols, routing protocols, and MAC protocols, as well as their interactions.

Beyond improvements to the operation of stand-alone WSN systems, there is a lot of traction around the integration of WSN into modern IoT architecture and platforms. The future of WSN lies in their proper integration with IoT systems. In this direction, WSNs will be increasingly combined with IoT-based cloud computing and edge computing systems, which will be presented in Chapter 4. Moreover, power-efficient and low-latency communications will be implemented not at the level of a WSN system only, but rather in the scope of more sophisticated IoT deployments that comprise multiple WSNs, along with other IoT devices. Hence, WSN deployment issues such as coverage maximization, lifetime optimization, and energy efficiency should be studied in the broader context of IoT applications [17].

Figure 2.3 depicts a typical integration of WSNs in a cloud or edge IoT application. It illustrates how entire WSNs can be integrated with an edge gateway via their sink nodes. Such edge or cloud architectures [18] will become more apparent in Chapter 4, where the convergence between IoT and edge or cloud computing will be discussed.

2.8 Summary

In this chapter, we introduced WSNs and their applications. We emphasized the design goals and challenges for WSNs, as well as the presentation of a reference architecture and the relevant layers of a WSN implementation. A specific part of this chapter was devoted to the importance and the functionalities of WSN middleware. Furthermore, concrete examples of WSN implementations were presented. We discussed popular WSN standards such as ZigBee, WirelessHART, and ISA100.11.a.

Despite the rising sophistication of IoT systems, WSNs are still among the most commonly used elements of IoT applications. It is therefore important for IoT researchers and practitioners to understand the basic elements of a WSN and the challenges that are associated with its design and integration in an IoT application. Some of these challenges are common to other types of IoT systems, as explained in the rest of the book. The future of WSN lies in their integration with other IoT components and technologies, including cloud computing, edge computing, fog computing, and a variety of other IoT devices. Most of these IoT technologies are presented later in the book. Prior to introducing these technologies, we present one more class of pervasive systems that supported the first generation of IoT systems and applications. Specifically, Chapter 3 presents RFID systems and their role in IoT deployments.

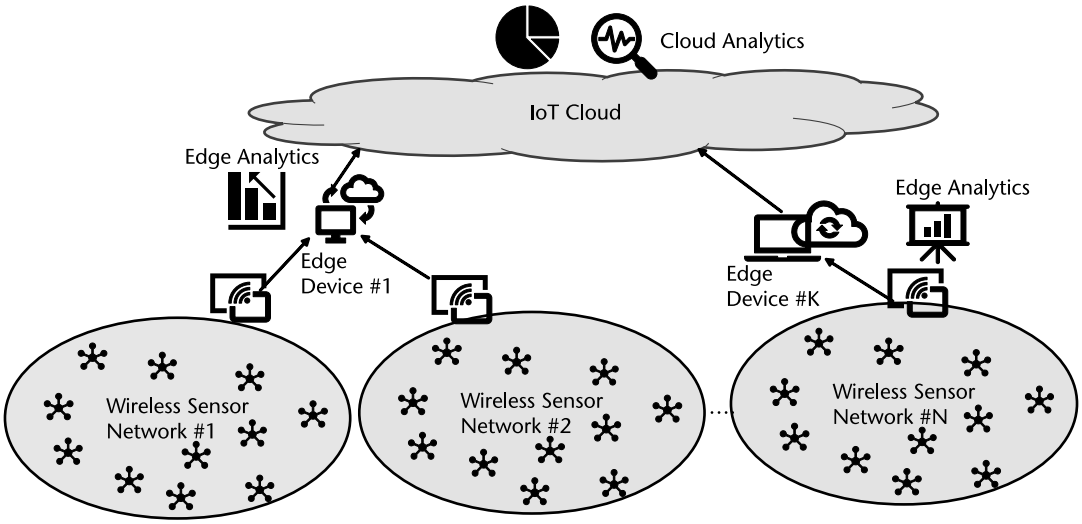

Figure 2.3 Integration of WSNs in a typical IoT (cloud or edge) architecture.

References

[1] Potdar, V., A. Sharif, and E. Chang, "Wireless Sensor Networks: A Survey," *2009 International Conference on Advanced Information Networking and Applications Workshops*, Bradford, 2009, pp. 636–641.

[2] Zhang, S., and H. Zhang, "A Review of Wireless Sensor Networks and Its Applications," *2012 IEEE International Conference on Automation and Logistics*, Zhengzhou, 2012, pp. 386–389.

[3] Ramson, S. R. J., and D. J. Moni, "Applications of Wireless Sensor Networks — A Survey," *2017 International Conference on Innovations in Electrical, Electronics, Instrumentation and Media Technology (ICEEIMT)*, Coimbatore, 2017, pp. 325–329.

[4] Kaschel, H., and J. Ortega, "Energy Efficiency in Routing Protocols Applied to WSN," *2016 IEEE International Conference on Automatica (ICA-ACCA)*, Curico, 2016, pp. 1–8.

[5] Russell, A. L., "OSI The Internet That Wasn't," *IEEE Spectrum*, March 2017.

[6] Akyildiz, I. F., et al., "A Survey on Sensor Networks," *IEEE Communications Magazine*, Vol. 40, No. 8, August 2002, pp. 102–114.

[7] Hadim, S., and N. Mohamed, "Middleware: Middleware Challenges and Approaches for Wireless Sensor Networks," *IEEE Distributed Systems Online*, Vol. 7, No. 3, March 2006.

[8] Khan, I., et al., "A Multi-Layer Architecture for Wireless Sensor Network Virtualization," *6th Joint IFIP Wireless and Mobile Networking Conference (WMNC)*, Dubai, 2013, pp. 1–4.

[9] TinyOS, http://www.tinyos.net/.

[10] Amjad, M., et al., "TinyOS-New Trends, Comparative Views, and Supported Sensing Applications: A Review," *IEEE Sensors Journal*, Vol. 16, No. 9, May 1, 2016, pp. 2865–2889.

[11] Di Felice, P., M. Ianni, and L. Pomante, "A Spatial Extension of TinyDB for Wireless Sensor Networks," *2008 IEEE Symposium on Computers and Communications*, Marrakech, 2008, pp. 1076–1082.

[12] GitHub, "LSIR/GSN: GSN Global Sensor Networks," 2020, https://github.com/LSIR/gsn.

[13] Aberer, K., M. Hauswirth, and A. Salehi, "A Middleware for Fast and Flexible Sensor Network Deployment," *Proc. of 32nd International Conference on Very Large Data Bases (VLDB '06), VLDB Endowment*, 2006, pp. 1199–1202.

[14] GitHub, "Gleicon/TinyMQ," 2020, https://github.com/gleicon/tinymq.

[15] International Society of Automation (ISA), "ISA100, Wireless Systems for Automation," https://www.isa.org/isa100/.

[16] Zhou, Y., Y. Fang, and Y. Zhang, "Securing Wireless Sensor Networks: A Survey," *IEEE Communications Surveys & Tutorials*, Vol. 10, No. 3, Third Quarter 2008, pp. 6–28.

[17] Priyadarshi, R., B. Gupta, and A. Anurag, "Deployment Techniques in Wireless Sensor Networks: A Survey, Classification, Challenges, and Future Research Issues," *The Journal of Supercomputing*, 2020.

[18] Shi, W., et al., "Edge Computing: Vision and Challenges," *IEEE Internet of Things Journal*, Vol. 3, No. 5, October 2016, pp. 637–646.

CHAPTER 3

Radio Frequency Identification

In this chapter, we introduce radio frequency identification (RFID) systems and technologies. Similar to WSN technologies, RFID technologies are among the forerunners of the IoT paradigm. They still play a significant role in modern IoT deployments. This chapter introduces RFID technologies, including the main elements of RFID systems such as tags and readers. We emphasize the introduction of the electronic product code (EPC), a standards-based identifier that is extensively used in the scope of RFID deployments. This chapter presents several standards that enable EPC RFID applications. Moreover, it presents popular applications of RFID technology, such as supply chain management applications. Finally, the importance of RFID middleware technologies and their role in nontrivial RFID deployments are also discussed.

3.1 The Basics of RFID Technology

3.1.1 RFID as a Forerunner of IoT

RFID is nowadays one of the most prominent automatic identification (AutoID) technologies. Many products are tagged with some RFID tag, which is used to identify them uniquely in the scope of applications such as warehouse management. RFID technology provides the means for transmitting the product's identification to another internet-connected system, which explains the affiliation between IoT and RFID [1].

RFID systems can be seen from different perspectives. The simplest perspective of RFID is its view as an identification technology (i.e., a technology for identifying objects and things tagged using RFID tags). This view defines RFID based on the RFID tag. A tag is a tiny chip comprising an antenna and a nonvolatile random access memory (NVRAM) for storing user data. Another view of RFID can be obtained if we consider not only the tag, but the RFID system as a whole. An entire RFID system comprises tags, but also RFID reader devices and middleware (i.e., system software). It enables wireless exchange of identification information about a tagged object and related user data, with other computing systems such as a server computer. This provides a foundation for using RFID tags not only in narrow scoped RFID applications, but also in a wider class of applications that comprise other types of internet-connected objects as well.

Another way to view a collection of integrated RFID systems is to consider them as a distributed database of object identifiers. In the scope of an integrated RFID deployment, there is usually a means of querying and obtaining information for a tagged object through its identifier. The type of information that can be obtained includes, for example, the location and the state of the tagged object. In this context, an RFID system can be also considered as an IoT database, which keeps track of "things"-related information. This is the main reason why RFID has been always considered as an instance of an IoT system. It is also the reason why RFID technology is considered a forerunner of the IoT paradigm.

The view of RFID as a forerunner of IoT is reinforced by the fact that RFID tags provide the simplest way for interconnecting objects to the internet. RFID tags can be considered as the most elementary pervasive computers, even though they are not Turing-complete machines (i.e., their computational capabilities are limited to practically nonexistent). RFID tags can be used to wirelessly connect objects to the internet. Likewise, any tagged object can be considered as an internet-connected object (ICO), which possesses the identify of its tag.

3.1.2 Operation and Components of an RFID System

To explain how RFID systems work, we first focus on the passive RFID tags, which are the most commonly deployed RFID systems. A simple RFID system comprises a reader and a tag, which are communicating over the air at a certain frequency. The reader has one or more antennas attached to it and hence an RFID system comprises readers, antennas, tags, and computing devices (Figure 3.1). An RFID solution uses a radio frequency (RF) signal to broadcast the data captured and maintained in the RFID tag, which is a programmable transponder comprising a chip.

Given these components or parts of an RFID system, the latter operates as follows:

- The reader sends out an electromagnetic wave at a specific frequency.
- The wave hits the RFID tag, and the tag then "scatters back" a wave at a different frequency with the chip's information encoded in those backscatter waves.

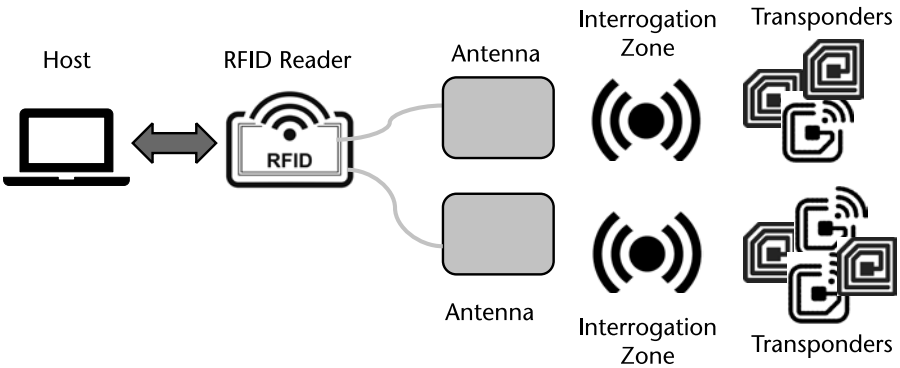

Figure 3.1 Basic operation of an RFID system using passive tags.

An RFID tag looks like a usual tag used on apparel items. However, there are many different types of RFID tags, which come in different shapes and morphologies to serve different applications. For example, a tag that is put on jewelry is different in shape and size from a tag used on clothing. This variety of tags is positive for supporting different applications. However, it also has a potential negative effect on the overall cost of a passive tag, as the existence of a variety of tags is a setback to the mass production of a single tag, which could have a very low price.

An RFID tag comprises a transponder chip and an antenna as shown in Figure 3.2. The RFID reader uses radio waves, which is a non-line of sight technology. Contrary to other popular AutoID technologies (e.g., barcodes), there is no need for line of sight between the reader and the tag, for the reader to read the tag. Furthermore, RFID technologies read tags at much longer distance than conventional RFID technologies (i.e., they have longer reading ranges).

RFID Tags come into different sizes and shapes. Moreover, RFID tags can be classified based on their energy autonomy. Hence, there are various RFID tag types, including:

- *Passive identity card:* Passive identity cards, or simply passive tags, contain only an ID (e.g., an EPC) in an unalterable form, along with cyclic redundancy check (CRC) information for transmission error detection. They are also referred to as a license plate.
- *Passive functional tag:* This is a broad category of tags that include any tag with functions over and above the functions of an elementary tag. Examples of such functions or features include user writable memory, sensors, and encryption.
- *Semi-passive tag:* This tag type comprises any tag that embeds battery technology to assist with providing power for the tag. In semi-passive tags the battery is not however the sole source of energy for the tag.
- *Active tag:* Active tags have a battery as their sole source of energy for the tag. Active tags have typically much higher reading ranges than passive tags. Therefore, they are used in a variety of applications that require large reading ranges, which are not provided by passive tags. Typical examples of such applications are localization and safety applications.

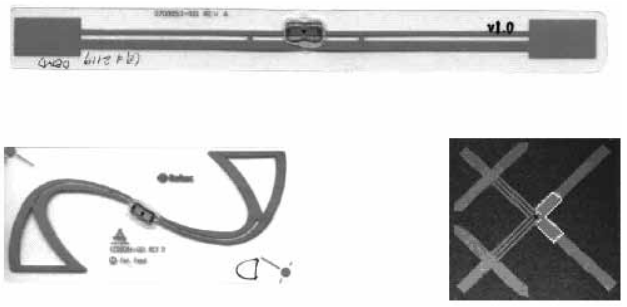

Figure 3.2 Different types of RFID tags.

Figure 3.3 illustrates a passive, a semi-passive and an active tag. The three tags comprise different components, which differentiate their capabilities in terms of energy autonomy and reading range.

A key element of every RFID deployment is an RFID reader. The RFID reader is a radio transceiver that picks up analog signals. The reader generates the signal that goes out through the antenna into space. It also listens for a response from the tag. Hence, an RFID reader receives analog waves and then turns them into bits of digital information. As already outlined, each reader is connected to one or more antennas (Figure 3.1). The deployment of more than one antenna boosts the accuracy of the readings. However, it is likely to lead to double readings of the same item. Double readings of the same time are filtered out in the system software (i.e. the middleware) of the RFID system.

3.1.3 RFID Numbers: The EPC

An electronic product code (EPC) is a specific example of an RFID identifier scheme, which is used in RFID tag. An EPC tag is therefore characterized by:

- The tag, including a chip, an antenna, and the packaging substrate;
- A numbering scheme that uniquely identifies all objects.

It incorporates existing EAN-UCC (European Article Numbering-Uniform Code Council) keys, and U.S. Department of Defense (DoD) constructs.

EPC tags are used to connect physical objects to computer networks. Figure 3.4 illustrates what an EPC code looks like. Its structure comprises a header, the EPC manager number, the object class, and the serial number of the product. An EPC is encoded on RF tags in bits (i.e., 0s and 1s). It comes in different formats, including various bit-length tags (i.e., 64 and 96 bits). This is done to accommodate existing identifiers and provide backward compatibility. No matter the format, unique EPCs are supported. EPCs are standardized by the GS1 organization (www.gs1.org), which defines standards and semantics for business data exchange. GS1 standards boost efficiency in many sectors, from retail and health to transport and logistics.

The basic EPC format comprises the following fields:

Figure 3.3 RFID tags: passive (left), semi-passive (center), and active (right) [2, 3]. (Reprinted with permission.)

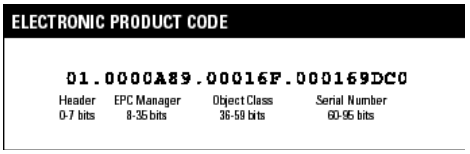

Figure 3.4 Structure of an EPC.

- *Header:* This identifies the length, type, structure, version, and generation of the EPC.
- *EPC manager number:* This is the entity responsible for maintaining the subsequent partitions.
- *Object class:* This identifies a class of objects.
- *Serial number:* This identifies the instance of the item and provides the means for supporting serialization.

Different EPC schemes are defined in the scope of different standards and organizations, including:

- A general identifier (GID) GID-96, which is a serialized version of the GS1 Global Trade Item Number (GTIN), SGTIN-96 SGTIN-198.
- The GS1 Serial Shipping Container Code (SSCC) SSCC-96, which is an 18-digit number used to identify logistics units. To automate the reading process, the SSCC is often encoded in a barcode (GS1-128 based), yet it can also be encoded in an RFID tag. It is used in electronic commerce transactions.
- The GS1 Global Location Number (GLN), SGLN-96 SGLN-195, which is used to identify a location and can identify locations uniquely where required.

Based on the mapping of different numbers, conversion from one number to another can take place in the scope of AutoID applications. For example, a conventional Universal Product Code (UPC) can be converted into a Global Trade Item Number (GTIN) through appropriate padding of the UPC with a 0 as indicator and another 0 number system carrier. In this way, a full 14-digit format, corresponding to the GTIN, can be built.

3.2 RFID Benefits, Opportunities, and Risks

3.2.1 RFID Versus Barcodes

RFID falls in the realm of AutoID technologies. The most popular and widespread AutoID technology is the conventional barcode, which is ubiquitous in supermarkets and retail stores. RFID introduced several properties that are advantageous over the conventional barcode. This is evident in the comparison of the two technologies across various properties. In particular:

- *Data modification:* The conventional barcode is read-only (i.e., it does not support modification). However, the passive RFID supports an unlimited number of modifications of the information that resides in its NVRAM.
- *Data security:* Passive RFID provides generally superior security features when compared to conventional barcode. Nevertheless, in some cases, RFID's higher reading ranges open up new security concerns, as adversaries could read information without being easily detected.
- *Capacity:* RFID offers higher data capacity (e.g., 64 kbps), while the capacity of barcode is limited to some tens of characters.
- *Cost:* Passive RFID tags are more expensive than barcodes. The cost of a passive tag can be approximately $0.10, but it cannot be competitive or comparable to the barcode's cost, which is typically below $0.01. This is one of the main and major advantages of barcodes when compared to RFID.
- *Reading distance:* The passive RFID provides a longer reading range (approximately 10m) when compared to barcodes that typically have reading ranges of no more than some tens of centimeters.
- *Life span:* A barcode has a more limited life span when compared to RFID.

A brief comparison between barcodes and RFID technologies is presented in Table 3.1.

Overall, RFID outweighs barcode in several properties, but the cost. However, the most important property of RFID that is a clear differentiator when compared to barcodes is serialization.

3.2.2 The Importance of Serialization

RFID was an AutoID technology that introduced serialization capability (i.e., the ability to identify individual items based on unique numbers [4]). Such serialization capabilities were not available in the early days of barcode technologies. Specifically, early barcodes identified the class of the product rather than the individual item. RFID technology and its associated number systems such as the EPC provided the means for implementing serialized applications that can track and trace individual products end to end. More recently, barcode technologies and numbering systems evolved to support serialization as well. For example, it is currently possible to implement serialization at the consumer unit level, based on the placement of

Table 3.1 RFID Versus Barcode Comparison

Property	*RFID*	*Barcode*
Data modification	Read/write	Read-only
Capacity	Higher (range of kilobytes)	Limited (tens of characters)
Cost	~$0.1	≤$0.01
Reading range	~10m	Approximately tens of centimeters
Life span	Higher/reusable	Limited/disposable

unique barcodes (e.g., two-dimensional (2-D) code or numeric codes) on the item. Likewise, existing standardized GTIN can be also used to enable barcode systems that provide serialization.

Serialization provides visibility on the entire supply chain and enables the implementation of end-to-end track and trace applications. In this way, it provides organizations with visibility into their operations across the entire supply chain. This enables value-added applications such as counterfeiting, as it becomes very easy to identify that a label of a product is fake or has been forged. Likewise, it provides a lot of efficiency gains in the supply chain, in operations like targeted withdrawal of entire lots of defected or problematic products. RFID technology was a pioneer in delivering these efficiency gains to enterprises.

Despite the distinct benefits of serialization, it is not always practical and cost-effective to tag individual items based on RFID tags. In such cases, it is also possible to tag palettes comprising multiple items, in order to leverage the benefits of RFID deployments at a palette level (Figure 3.5). In this case we usually refer to palette-level tagging. On the contrary, when tagging individual items, we refer to item-level tagging. In general, serialization is sometimes considered a quite expensive solution, which is not suitable for smaller companies (e.g., small medium enterprises (SMEs)) and organizations that work in relatively small geographic markets.

3.2.3 Benefits of RFID Applications

Based on the properties of RFID systems, RFID delivers the following benefits:

- *Serialized data and traceability:* Through serialization, RFID systems can keep accurate account of items and their properties. RFID deployments provide full context regarding the tracked and traced items. In this way, RFID applications provide enhanced traceability information.
- *Real-time data collection and speed:* Given that RFID does not require line of sight for its operation, RFID systems are able to reduce human intervention, as readers can read multiple items in each read cycle. Hence, RFID can increase the throughput and speed of supply chain operations.
- *Security benefits:* RFID tags provide enhanced security and facilitate secure information exchange. RFID's security functionalities can be used in conjunction with the serialization properties to enable counterfeiting prevention

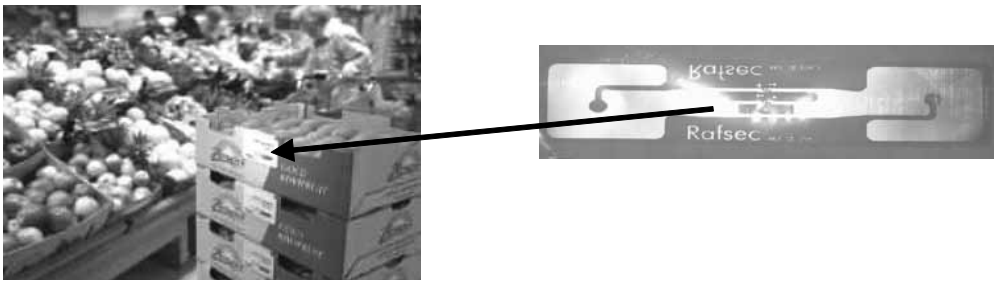

Figure 3.5 Palette-level tagging.

in industries where fake products are commanly found (e.g., in apparel and pharmaceuticals).

- *Added-value services:* RFID tags can be enhanced with additional sensing features (e.g., temperature sensing, Global Positioning System (GPS) sensing). In this way, RFID can enable added-value applications and services such as cold chain management (e.g., tracking the temperature of goods across the supply chain) and location-based services that use the location of tagged items.

3.2.4 From RFID Retail Deployments to Amazon GO

The first large-scale RFID deployments date in 2005 to 2006, where large retailers such as Walmart automated several of their supply chain processes based on RFID technology. The METRO Group's GALERIA Kaufhof department store in Germany [5] was among the first to implement a breakthrough item-level tagging retail apparel deployment in 2007. The implementation was a milestone for the retail industry, delivering consumer-facing RFID applications integrated from the distribution center to retail smart shelves and cashier checkout stations. Around 30,000 items in the menswear department were tagged with RFID transponder chips. RFID readers installed in the receipt area, at all transition points, as well as inside the dressing rooms and at the checkout desk, read the EPC without requiring physical or visual contact. Moreover, the deployment of RFID enabled a wide range of added-value services. For example, the smart dressing rooms and smart shelves in the retail stores were providing special services for the customers based on smart dressing rooms that comprised shelves and mirrors equipped with RFID readers. The latter were able to read the RFID tag of an apparel item to instantly check whether the item was available in stock. Furthermore, suggestions about other apparel items that can be used to match the given one were presented to the consumer.

Overall, many different ideas can leverage the identification capabilities provided by RFID technology, in order to enhance the efficiency of business processes (e.g., checking the availability of given items in specific sizes and colors), reduce costs, and offer the consumer with individualized services. Hence, the deployment of RFID technologies has been a milestone that advanced the modernization process in the retail sector.

In general, RFID deployments in the retail sector established and promoted the vision of the "store of the future," which provides features such as:

- Fast and accurate inbound goods receipt;
- Back-room, real-time inventory management;
- Fixed and handheld readers tracking real-time sales floor inventory;
- Smart mirrors showing complementary clothing choices or accessories;
- Smart shelves with monitors indicating available garment size and style choices;
- In-aisle product information triggered by scanning items;

- RFID-enabled, point-of-sale terminals delivering efficient checkout.

The "store of the future" vision was articulated more than 10 years ago. Nevertheless, it was never deployed at scale based on RFID technology. One of the reasons for this was the high cost of the RFID labels, which become an obstacle to the wider adoption of RFID in the retail sector. However, the "store of the future" concept has evolved and matured over the years. Nowadays, Amazon has built and operates its Amazon Go convenience stores in the United States. The stores are partly automated and enable customers to purchase products without a need to be checked out by a cashier. These Amazon stores implement a significant part of the "store of the future" concept, as they automate the purchase, checkout, and payment steps of retail transactions. Amazon stores are not based on RFID technology. Rather, they utilize several state-of-the-art technologies, such as machine vision, deep learning, and the fusion of data from multiple sensors. Hence, the deployment of RFID technologies in retail can be considered a forerunner to the revolutionary concept of Amazon Go stores, since they established a vision for streamlining supply chain management, inventory management, and the customer experience.

3.2.5 RFID in the Current Industrial IoT Era

RFID is currently used in many IoT applications, beyond retail and supply chain management use cases. For instance, RFID tagging enables flexibility in production lines as part of Industry 4.0 deployments. Specifically, production lines are commonly equipped with RFID readers. The latter are used to read RFID-tagged products and to customize automation functionalities accordingly. Hence, RFID technology can support cost-effective massive customization in modern production facilities. As another example, RFID technologies are used to support the development and deployment of authentication and authorization functionalities in the scope of industrial IoT systems. Furthermore, active RFID tags are used in long-range, track-and-trace applications such as asset management and worker safety applications.

Overall, RFID systems and technologies are among the most common elements of state-of-the-art IoT systems. In today's industrial IoT era, there are many sophisticated and versatile IoT systems which comprise RFID technologies. This is one of the main differences between early RFID deployments in the beginning of the twenty-first century and the current sophisticated industrial IoT deployments. Another major difference lies in the fact that early RFID deployments were mostly passive (i.e., focused on information collection, processing, and presentation). Nowadays, RFID systems are integral parts of various CPS that are used to drive actuation and control functionalities.

3.2.6 RFID Privacy Challenges

RFID technologies did not only unveil a great deal of business opportunities in sectors like retail, trade, and industry. They also introduce new privacy and data protection challenges. For example, reading RFID tags after the sales of a retail

item provides the means for invading an individual's privacy, through revealing purchases, preferences, and other aspects of the consumer's personal life. This is the reason why every RFID application should be accompanied by a privacy impact assessment (PIA) [6], which is destined to identify the privacy implications of an RFID application, along with measures that can alleviate these implications [7]. At the technical and technological levels, there are already several solutions that can support privacy-friendly operations. Typical examples are RFID tags on passports that become readable only when the passport is opened, as well as configurable tags for retail items that can be deactivated after an item is sold. Note also that tags in retail products could be also removed after purchase. However, in the removal case, buyers will not be offered with the opportunity of enjoying RFID-enabled after-sales services (e.g., fast returns).

3.3 RFID Middleware

3.3.1 Drivers and Motivation

The processing of identifiers, the filtering of tags, and the generation of events (e.g., such as that an object was removed from a supermarket shelf) are all performed in the scope of special system-level software, which is called RFID middleware. As discussed in earlier chapters, IoT technologies such as WSNs need system-level software to derive and properly integrate the context about the things to business applications. RFID technology is no exception. Indeed, there are some very good reasons why RFID middleware is needed [8, 9].

First, in an RFID application, excess information must be filtered out. You can think of repeatedly reading tags that do not add information for the business application at hand. A similar situation appears when the same tag is read and reported from more than one reader or antenna. Filtering is therefore needed to get rid of information that is redundant and hence not useful. Filtering is essential towards making wise use of network bandwidth, but also towards avoiding confusing information inside business applications. As a result, system-level software is needed towards applying and implementing filtering of excess information.

Second, middleware is needed in applications that comprise two or more readers that "do not speak the same language." RFID deployments may be supported by more than one type of reader or even readers from multiple vendors. Such readers come with different APIs and hence the implementation of custom integration logic for each reader vendor is required, which is tedious and time-consuming. RFID middleware can ease the interfacing to different readers.

Third, RFID middleware is related to the need to pass information derived from an RFID deployment (e.g., several readers and antennas) to different applications and data stores. RFID information is typically routed to multiple enterprise applications and databases, such as enterprise resource planning (ERP) systems, manufacturing resource planning (MRP) systems, and warehouse management systems (WMS). RFID middleware facilitates the routing of RFID information to different enterprise applications and databases.

3.3.2 RFID Middleware Functionalities

RFID middleware provides, in principle, four functionalities:

- *Reader and device management:* This is, for example, middleware for the problem of interfacing to multiple readers from different vendors.
- *Scalable application integration:* This is the functionality needed to route information to different enterprise systems.
- *Data and process management:* This is the functionality needed to filter data values and produce events as required by different applications.
- *Partner integration:* This is the functionality needed for integrating the RFID system with applications of partners (i.e., third-party applications).

Some examples of RFID middleware functionalities in each one of these categories are:

- *Reader and device management:* Typical functionalities include configuring, deploying, and issuing commands directly to readers (e.g., to turn off a reader). RFID middleware implementations provide a common interface to different readers regardless of their vendor.
- *Data management:* Typical data management functionalities include the intelligent filtering of data to get rid of excessive information, as well as the routing of data to appropriate destinations. Although filtering and data routing sound like easy and straightforward functionalities, they involve low-level logic and complex algorithms.
- *Application integration:* Application integration typically involves the implementation of messaging, routing, and connectivity features to reliably integrate RFID data into existing supply chain management (SCM), ERP, WMS, or customer relationship management (CRM) systems. RFID application integration can leverage a service-oriented architecture (SOA) to enable a loosely coupled approach to integrating different systems, including the RFID system and the enterprise systems outlined above.
- *Partner integration:* Typical RFID middleware functionalities for partner integration involve sharing RFID data with partners towards improving collaborative processes such as demand forecasting and vendor-managed inventory. RFID middleware for partner integration implements business-to-business (B2B) integration features, such as data exchange compliant to the Electronic Data Interchange (EDI) Protocol. The latter is extensively used in e-commerce applications. An RFID system captures information about products automatically and reliably, while the RFID middleware undertakes to convey the information to business partners' systems in an EDI-compliant format.
- *Process management and application development:* Typical RFID middleware functionalities for process management and application development involve the orchestration of RFID-related processes end to end. Process

management applications are likely to deal with multiple applications and multiple enterprises.

- *Scalability and administration:* RFID middleware for scalability and administration of the RFID data management infrastructure provides the means for reliable processing of huge amounts of data. As part of a scalable architecture, RFID middleware can be used to balance processing loads across multiple servers and to automatically reroute data when needed (e.g., when a server fails). RFID deployments can be parts of wider big data deployments, since RFID tag readings can contribute large volumes of data with very high ingestion.

3.3.3 RFID Middleware Architectures

RFID middleware functionalities are structured and integrated in the scope of enterprise applications. Some of the already discussed RFID middleware functionalities are implemented close to the readers. For example, this is the case with RFID tag-filtering functionalities. Some other functionalities are implemented at the back end of a system implementation, where enterprise systems and applications reside.

Most RFID middleware implementations follow a typical three-tier architecture, which places middleware functions in three distinct yet complementary tiers:

- *Edge tier:* This implements functionalities that are close to the field (i.e., close to the physical world). The merit and need of such a tier for IoT applications are discussed in more detail later in this book. Specifically, in Chapter 4, we introduce edge computing and the deployment of IoT as part of edge computing architectures.
- *Operational tier:* This implements the mapping of functionalities that convert low-level functionalities (e.g., tag-level processing) to business-level functionalities (e.g., business events). In a sense, this is the tier converting the low-level semantics of RFID information to higher-level business semantics.
- *Enterprise tier:* This deals with the routing and integration of business-level (or enterprise-level) RFID information to enterprise systems and applications (e.g., ERP, WMS).

The multitier architecture of RFID systems is illustrated in Figure 3.6, which presents the three tiers outlined above, along with the field tier where the RFID-tagged objects reside.

This multitier RFID middleware architecture is rather high-level. A detailed architecture may entail many more layers and middleware functionalities. Nevertheless, this multitier architecture provides a conceptual model that facilitates the classification of middleware functionalities based on their physical and logical placement in the scope of an RFID deployment. It can be used as a reference when discussing different middleware functionalities, since it enables the classification of RFID middleware functionalities according to the tier that they are deployed.

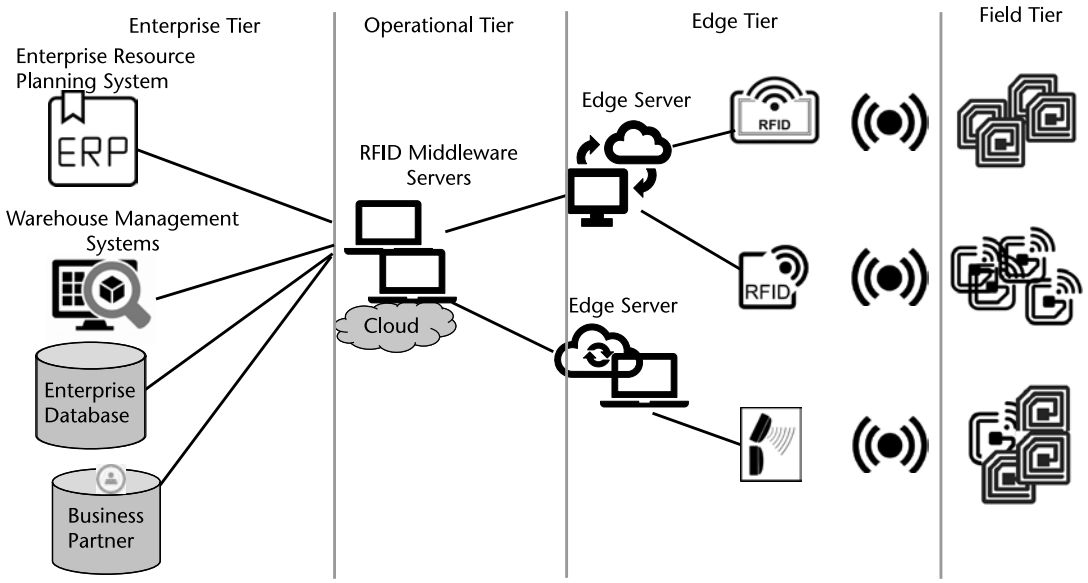

Figure 3.6 Multitier architecture of RFID deployments.

3.4 The EPC Architecture Framework

3.4.1 Introduction

GS1 is not only standardizing the format of the EPCs. It also provides a standardization of the structure of an entire RFID system. This includes the way that information is captured from an RFID system in an enterprise and the way that it is communicated to other parts of it. By providing standards about the reading, filtering, and, ultimately, the exchange of RFID/EPC information across enterprise systems and data stores, GS1/EPCglobal is also providing guidelines for relevant RFID middleware implementations. Figure 3.7 provides a high-level overview of the EPCglobal Architecture Framework, including relevant GS1/EPC standards. In particular, the EPC architecture framework specifies three categories of standards:

- *EPC Physical Object Exchange Standards:* These deal with the specifications of the EPC schemes, along with mechanisms for exchanging physical objects with EPCs.
- *EPC Infrastructure Standards:* These deal with the specification of the infrastructures that enable the registration, lookup, and other operations that are essential to the functioning of the EPC systems within EPC subscribers (i.e., companies that subscribe to GS1).
- *EPC Data Exchange Standards:* These provide the standards that enable the interactions between different parties across EPC-enabled supply chains.

A more detailed view of the GS1 copyrighted EPC architecture framework, including specific components and modules that comprise an RFID deployment

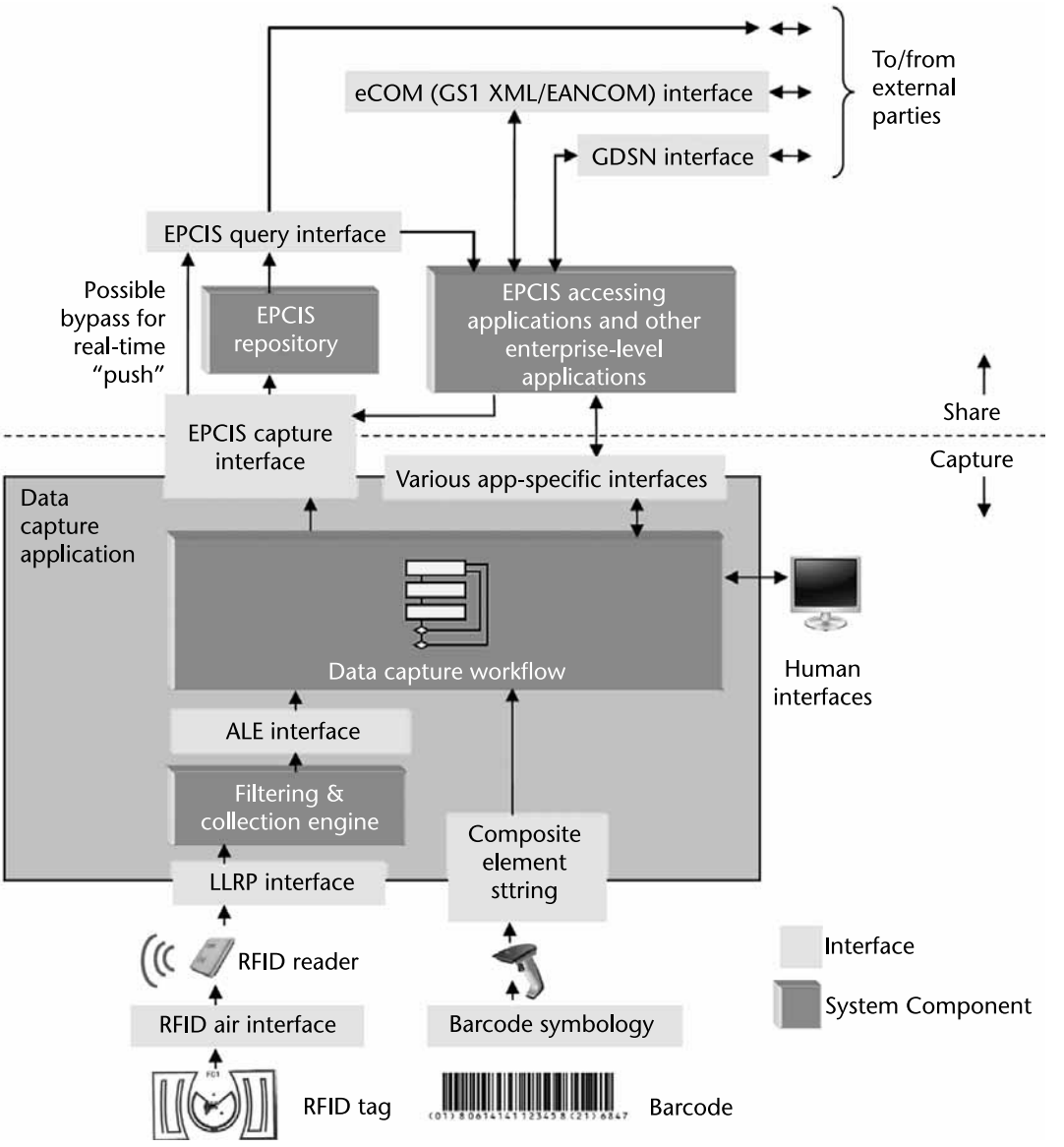

Figure 3.7 Overview of the EPC architectural framework [13, 15]. (Copyright GS1. Reprinted with Permission.)

and the interfaces between them, is provided in Figure 3.7. The main middleware modules can be summarized as follows:

- *RFID tag:* This is an EPC tag, which can be read by an RFID reader as specified in GS1/EPC standards like the TagProtocol.
- *RFID reader:* This represents an RFID reader device that reads the tags. It is accessible from higher-level middleware functions based on GS1's/EPC Reader Protocols.

- *Reader management:* It represents an application for managing (e.g., configuring) an RFID reader. The management interactions between reader management and RFID reader are specified in a reader management interface.
- *Filtering and collection engine (RFID middleware):* This is an RFID middleware module, which interfaces to RFID readers and implements filtering functionalities. It exports and provides information to higher layers of RFID middleware based on an application-level event (ALE) interface [10]. Through this interface, other middleware modules and applications access application events rather than raw tag streams [11].
- *EPCIS capturing application:* This is an application that leverages ALEs and converts them to business-level events, formatting according to the EPCIS (EPC Information Sharing Standard) [12] that is discussed later in this chapter.
- *ECPIS repository:* This repository can be regarded as a database of business-level events, which follow the EPCIS standard. An EPCIS Capture Interface is used to capture and access such events from the EPCIS capturing application and to store them in the repository [10, 12].
- *EPCIS accessing applications:* These are applications integrated within enterprise systems (e.g., ERP, WMS) that use an EPCIS Query Interface to capture information from the EPCIS repository. Through such a query interface, EPCIS capturing applications can access full context about tag objects, such as where and when they were "seen" (i.e., read), as well as in the scope of which purpose. This is because the business events provide tag information enriched with contextual information as specified in the EPCIS standard. EPCIS Accessing Applications may reside either within the enterprise where the EPC RFID system is deployed (intra-enterprise), but outside this enterprise (i.e., from another enterprise).
- *Local ONS (Object Naming Service):* It is possible for an enterprise to operate an ONS (specified in EPC/GS1), which provides naming and directory services for tags and tagged objects that are stored in EPCIS repositories. The concept is like the Domain Name Service (DNS) directory in the scope of the internet, which provides the means for looking up internet addresses. However, in this case, one can look up tagged objects (i.e., EPCs) to get information about the repositories that comprise information about the specified tagged object.

The EPCglobal Architecture Framework facilitates the understanding of the various building blocks of a complete RFID middleware system. Nevertheless, from a practical perspective, it has been deprecated and its contents are subsumed into the more recent GS1 System Architecture document [14] (also copyrighted by GS1).

3.4.2 EPC RFID Reader

The most fundamental element of an RFID deployment is the RFID reader. An EPC-compliant RFID reader is a reader able to read EPC tags. It is not merely an interrogator, but rather it comprises the following:

- Read points (i.e., antennas or barcode scanners);
- Sources, which are read-point concentrators.

The basic functions of a reader include:

- Detecting RFID tags within its electromagnetic (EM) field (i.e., a query function in the reader's range).
- Repeating query sessions periodically based on a triggering event basis (EPC readers do not depend on external triggers; rather they are internally triggered and programmed to repeat the querying function);
- Collecting the results of multiple query session, filtering, and reporting them (hence, readers comprise filtering and reporting functionalities);
- Implementing a message transport binding.

Readers can be combined in various configurations. As physical devices, they can be distributed across locations as shown in Figure 3.8. The readings (i.e., reports) of the various readers can then be collected and processed in a back-end application. Different middleware modules for one or more readers are typically implemented. The deployment of multiple physical readers and the collective processing of their readings can be characterized as a (logical) distributed reading functionality (i.e., a distributed reader).

3.4.3 EPC RFID Middleware

The RFID middleware component of the EPC architecture is based on the EPC-ALE and EPCIS [15] specifications. It is the middleware module that undertakes to collect tag readings from multiple physical readers and performs the following functions [10, 16]:

- *Data collection and filtering:* This includes source abstraction (i.e., at the level of the ALE middleware, the actual sources are abstracted, and applications see only logical readers).
- *Space multiplexing and time multiplexing:* This is performed through aggregating and unifying tag readings from multiple readers in both space and time. The RFID middleware implements the ALE specifications. It reads tags across readers and event cycles within these readers.

The event cycle of a reader defines the interval where a report is produced by a reader. Likewise, the ALE cycle defines the time boundaries of the interval during which the middleware receives reports from multiple physical readers comprising

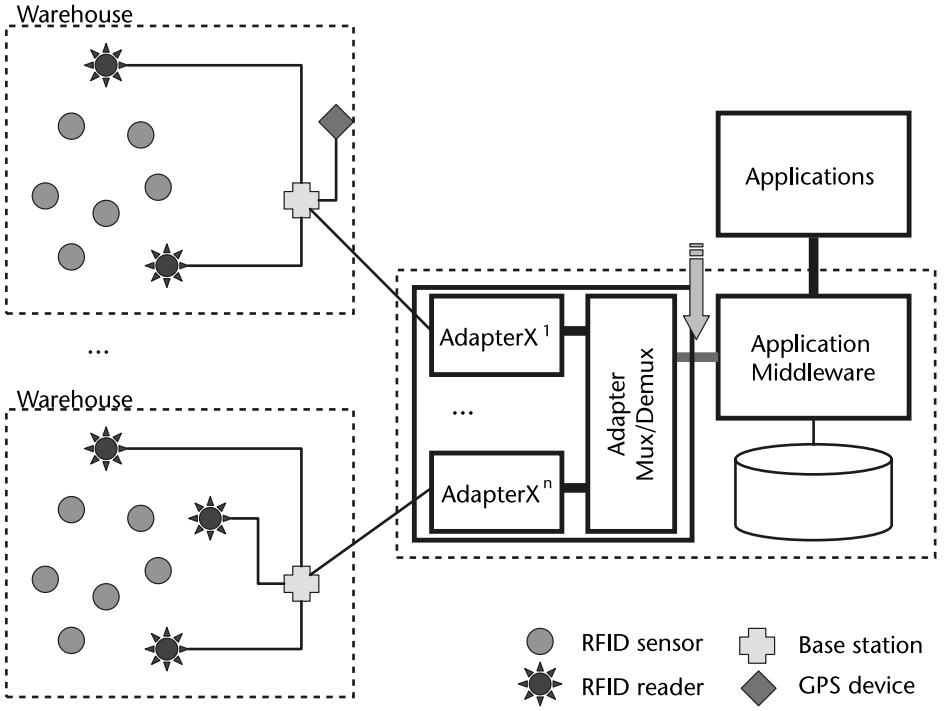

Figure 3.8 The concept of a distributed (logical) reader.

a logical reader. The RFID middleware will collect the reports produced by the various readers (comprising the local reader) within the specified boundaries. The RFID middleware can be configured in terms of the ways it filters and processes the reports, as well as in terms of the ways the various physical readers produce their reports (e.g., time-based, event triggered). The latter configurations are standardized within an appropriate configuration file called the EC specification. Hence, the RFID middleware module is configured on the basis of the EC specification, which can be assigned to an instance of the RFID middleware programmatically.

Readers interact through the RFID middleware (i.e., ALE engine) with the applications consuming ALEs. These ways include both poll modes where the ALE consumer asks the ALE engine for a report and push modes where the ALE consumer subscribes to the ALE engine to receive reports periodically or upon the occurrence of specific events. In particular:

- *Subscription mode:* The ALE consumer application submits a configuration file to the ALE engine, which specifies the desired content of RFID tag reports. This configuration file will specify the events of interest to the ALE consumer. Based on this configuration file, the ALE engine will submit reports containing the specified content (i.e., specified event types) to the consumer based on a callback mechanism.
- *Polling mode:* In this mode, the ALE consumer submits a configuration file that specifies the desired content of the RFID tag reports (e.g., events of

interest). Accordingly, it can request reports to the ALE engine and receive reports based on the specified format. The difference from the previous mode is that the reports are not submitted automatically based on some subscription and callback mechanism. Rather, they are submitted whenever requested by the consumer.

- *Request response mode:* This mode involves a request-response paradigm, where the ALE consumer specifies the format of a custom report, which is accordingly prepared and submitted from the ALE engine back to the ALE consumer.

3.4.4 EPCIS: Sharing EPC Information

EPC applications end up modeling and presenting RFID information based on the EPCIS standard. This standard specifies four event types:

- *Object event:* This represents EPC-tagged objects.
- *Aggregation event:* This represents aggregations of EPC tags, such as a collection of EPC tags that are aggregated within an EPC-tagged container.
- *Transaction event:* This denotes a (business) transaction occurring in the scope of EPC applications.
- *Quantity event:* This represents the appearance of multiple items (e.g., barcode items) in the scope of a transaction.

Each of these events comprises several attributes, which are defined based on the following semantics (EPCIS semantics):

- *EPC number:* This represents the tag "read" in the application (i.e., it denotes what was seen in a retrospective fashion).
- *Parent EPC:* This represents the EPC of an item (e.g., tagged container), which contains other EPC tags (i.e., denotes an aggregation).
- *Read point ID:* This identifies the point where the EPC item was read (i.e., denotes where the tag is in a retrospective fashion). A read point is a discretely recorded location meant to identify the most specific place at which an event took place.
- *Business location ID:* This specifies a business location. A business location is a uniquely identified and discretely recorded location meant to designate the specific place where an object is assumed to be following an EPCIS event until it is reported to be at a different business location by a subsequent event. It defines the prospective location of an EPC tag.
- *Business step ID:* This is associated with an event and specifies its business context, that is, the business transaction in the scope of which the EPC tag is read (e.g., shipping). It defines why an EPC tag was read in retrospective.

- *Disposition ID:* This specifies the business condition of the event's objects, subsequent to the event. It also gives information about why an EPC was seen, but in perspective.
- *Event time:* This is the when of a transaction. This is the timestamp that accompanies any EPCIS event.

Figure 3.9 depicts the various attributes of EPCIS events in the form of a class diagram. All the four events inherit the attributes of the EPCIS event. Object events, aggregation events, transaction events, and quantity events have then their own specific attributes, which were specified previously. There is also a BizTransaction class, which denotes the list of different business transactions that take place in the scope of the EPC application. In the scope of an application, the observed events (e.g., object events) can correspond not only to a single EPC, but rather to a list of several EPCs. At the same time, an aggregation event comprises not only a list of EPCs, but also the parent EPC, which defines the aggregation (i.e., the list of EPCs in an aggregation event denotes the list of EPC-tagged items that are contained in the EPC-tagged item denoted by the parent ID). All the different events are associated with read points, business locations, and business steps, according to the already presented semantics.

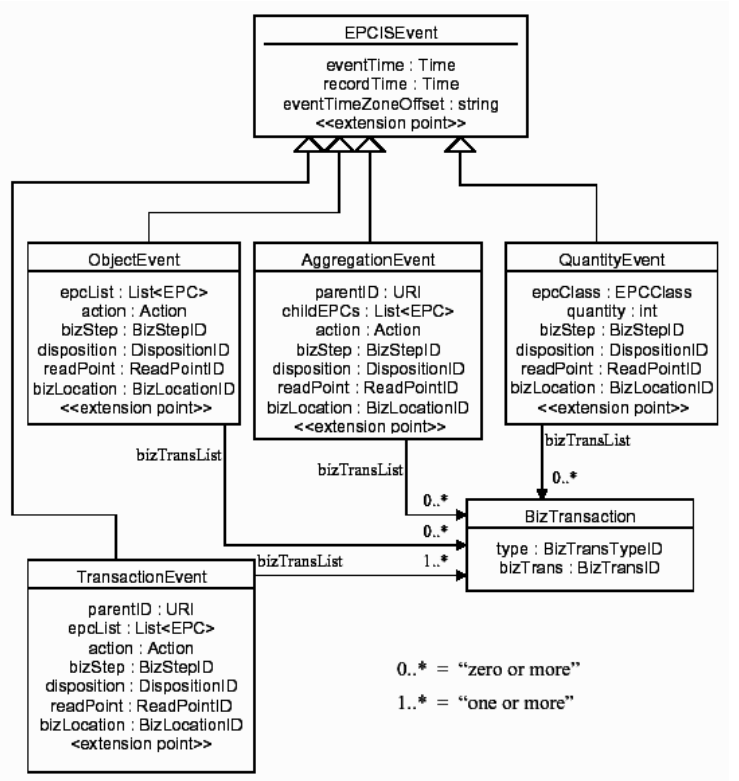

Figure 3.9 Taxonomy of EPCIS events [15]. (Copyright GS1. Reprinted with permission.)

EPCIS events can be stored in a database. Based on a trail of the events in a database, one can implement several RFID applications, such as traceability applications.

3.4.5 EPCIS: A Warehouse Management Example

EPCIS applications depend on a proper modeling of the business setting, based on EPCIS semantics. The modeling of the business setting is typically based on data that do not frequently change over time, which are characterized as master data. They represent the metadata of the deployment. As an example, we model a company in terms of its warehouse management operations. We assume that the company possesses various logical spaces identified as warehouses. The nth warehouse is denoted as W_n ($n = 0, 1, 2, \ldots$)). Warehouses are organized in a hierarchical manner so that each warehouse is contained within another warehouse.

All warehouses can be collectively aggregated under warehouse W_0, which can be considered a physical central warehouse or the company itself. Child logical warehouses may correspond to physical warehouses or other units of storing capacity down the hierarchy. For example, shelves contained within a physical warehouse space can be logically considered as child logical warehouses, which can also contain packages that can also be considered logical warehouses down in the hierarchy and so on.

Warehouse management processes are based on tagged containers. The nth container is typically denoted as C_n ($n = 0, 1, 2, \ldots$)). Containers of different types can be used, such as pallets, carton boxes, and carts. Containers are organized in a hierarchical fashion, which allows containers (e.g., pallets) to contain other containers (e.g., carton boxes). A container is situated in a parent logical warehouse. Moreover, a container (C_n) is contained in a warehouse, as soon as this warehouse contains a parent container of C_n (i.e., the hierarchy is respected).

Both containers and logical warehouses can contain other containers and/or items. The key differences are:

- When items within a container move, the container moves as well.
- When items within a logical warehouse move, the logical warehouse does not move.
- One logical warehouse typically has one parent object (i.e., the parent warehouse).
- A container typically has two parent objects (i.e., a parent warehouse and a parent container).

Given this modeling various RFID-based warehouse management processes can be represented based on appropriate EPCIS events. As a first step, we consider basic (elementary) warehouse management processes, such as receiving, moving (of items) with warehouses, pick and pack, order shipment, and inventory processes.

Each of the above processes is associated with various business events according to the EPCIS standard. For example, in the scope of a shipment process, goods

are delivered to the warehouse and relevant EPCIS events are produced in the warehouse management systems to document this process. The following points are essential to understanding this process:

1. The business process takes place in the scope of the shipping warehouse (W_S).
2. The products that must be shipped are assembled. The process involves moving items and containers (with items) out of warehouse W_S. It is assumed that containers have been put within carts during order collection.
3. During shipment, these aggregations are deleted. Items and containers are moved out of the pick-and-pack carts. New aggregation events signify the creation of packing lists for the shipment process. Objects are moved out of the warehouse.
4. Transaction events are issued to convey and control the status of the process. Transaction-observed events provide insight into the objects that have been shipped. A transaction finish event is issued to signify the end of the project.
5. The system can finally check automatically whether the packing list comprises the same products as the shipment list.

Based on the definitions and assumptions that we made regarding the warehouse management processes, Figure 3.10 presents an aggregation EC-IS event, which is produced when products are moved out of a cart (C_n). The action associated with this event is "delete," which means that a preexisting aggregation is deleted, or that, following the production of this event, the aggregation will no longer exist in the database. The <EPC List> denotes the list of EPCs (products) that are moved out of the cart. The parent EPC field is the container C_n, which is the tagged container that comprises the EPCs. Figure 3.11 illustrates the event that is associated with the packaging of the items in container C_n. The action associated with this event is "add," which denotes that an aggregation is created.

Event Description			
EventType	Time	bizStepID	dispositionID
AggregationEvent	Time	Null	Null
bizLocationID	readPointID	EPC	parentEPC
Null	Null	<EPC List>	C_n
Action	bizTransactionTypeID	bizTransactionID	
Delete	Null	null	

Figure 3.10 Aggregation event denoting that objects are moved out of the cart (Cn) (i.e., aggregation deleted).

Event Description			
EventType	Time	bizStepID	dispositionID
AggregationEvent	Time	null	null
bizLocationID	readPointID	EPC	parentEPC
W_S	W_S	<EPC List>	C_n
Action	bizTransactionTypeID	bizTransactionID	
Add	Null	null	

Figure 3.11 Packaging of objects within a container (Cn).

Figure 3.12 illustrates an aggregation event corresponding to the event of moving a whole group of packaged objects (i.e., objects packaged in container C_m) out of a cart represented by C_n. The fact that the objects are moved out is denoted by the Action field, which in this case is again "Delete." This event shows the versatility of expressing the movement of whole containers or carts as part of warehouse management processes.

Figure 3.13 illustrates an object event, which is associated with the action "delete." It shows objects leaving the warehouse W_s, which is the identifier of the warehouse that is set at the readPointId field. The objects that leave the warehouse are listed in the <EPCList>.

Figure 3.14 lists a transaction event, which is associated with the action "observed." It is issued in the scope of the warehouse management processes in order to illustrate the context of the business transaction in the scope of which other events occur. The transaction event is associated with a list of products (EPCs), which are given as part of the <EPC List>. It is also associated with a business transaction (BT_n in this case), which denotes the wider business transaction that comprises this transaction event. Moreover, the bizStepID and dispositionID are also set (to values D_m and D_n in this case) in order to denote the status of the

Event Description			
EventType	Time	bizStepID	dispositionID
AggregationEvent	Time	Null	null
bizLocationID	readPointID	EPC	parentEPC
Null	Null	C_m	C_n
Action	bizTransactionTypeID	bizTransactionID	
Delete	Null	Null	

Figure 3.12 Aggregation event denoting that a whole group of objects (Cm) (e.g., package) are moved out of the cart (Cn) (i.e., aggregation deleted).

Event Description			
EventType	Time	BizStepID	dispositionID
ObjectEvent	Time	Null	Null
bizLocationID	readPointID	EPC	ParentEPC
Null	W_S	<EPC List>	Null
Action	bizTransactionTypeID	bizTransactionID	
Delete	Null	Null	

Figure 3.13 Objects leaving the warehouse WS where the shipment is conducted (i.e., object event delete).

Event Description			
EventType	Time	bizStepID	dispositionID
TransactionEvent	Time	D_m	D_n
bizLocationID	readPointID	EPC	parentEPC
Null	Null	<EPC List>	Null
Action	bizTransactionTypeID	bizTransactionID	
Observed	Null	BT_n	

Figure 3.14 Transaction event for objects that have been shipped.

objects at the time when this transaction events are issued. D_m denotes the current step of the business transaction BT_n, while D_n is the next step in the same transaction. This is very useful as it allows applications to track the steps that are entailed in the business transaction.

Like all other events, this event is associated with a timestamp (at the Time field), which denotes when this event was issued.

Figure 3.15 shows another transaction event, which is used to denote that the shipment process was concluded. Contrary to the previous event, which was associated with the action "Observed," this one is associated with the action "Delete." This means that the transaction is concluded based on this event. The event is also associated with a list of EPCs (products) listed in the <EPC List>. It is also associated with the bizStepID D_n, which, based on the previous event, was the next step in the ongoing (i.e., "Observed") BT_n transaction. In the case of this transaction event, the dispositionID is set to NULL, given that there is no next step in the business transaction BT_n.

Event Description			
EventType	Time	bizStepID	dispositionID
TransactionEvent	Time	D_n	Null
bizLocationID	readPointID	EPC	parentEPC
Null	null	<EPC List>	Null
Action	bizTransactionTypeID	bizTransactionID	
Delete	Null	BT_n	

Figure 3.15 Transaction event for concluding the order shipment process.

Overall, EPCIS events are issued as RFID tags are read in the scope of a business process. Issued events are persisted in the EPCIS repository (i.e., a database) and can be used by RFID and IoT applications to implement functionalities such as traceability, management of business processes, business intelligence, and data analytics. The events are issued and persisted to a database by RFID middleware, notably middleware that processes ALEs and enhances them with the required semantics in order to convert them to EPCIS events. The master data (e.g., information about warehouses, read points, transactions) provide the means for adding meaningful context to the RFID readings. They also enable the conversion of RFID tag streams to a series of EPCIS events that explain the what, why, where, and when of an RFID process.

As already outlined, an end-to-end supply chain or warehouse management process is likely to entail multiple business transactions. As soon as these are RFID-enabled, each one of them can be associated with multiple EPCIS events (Figure 3.16). By persisting the issued events in EPCIS repositories (databases), one can track and trace entire business processes. The monitoring and management of a composite business process that comprises multiple transactions may entail multiple EPCIS repositories, owned and operated by different companies across the supply chain. GS1 provides standards for dynamically locating the repositories that

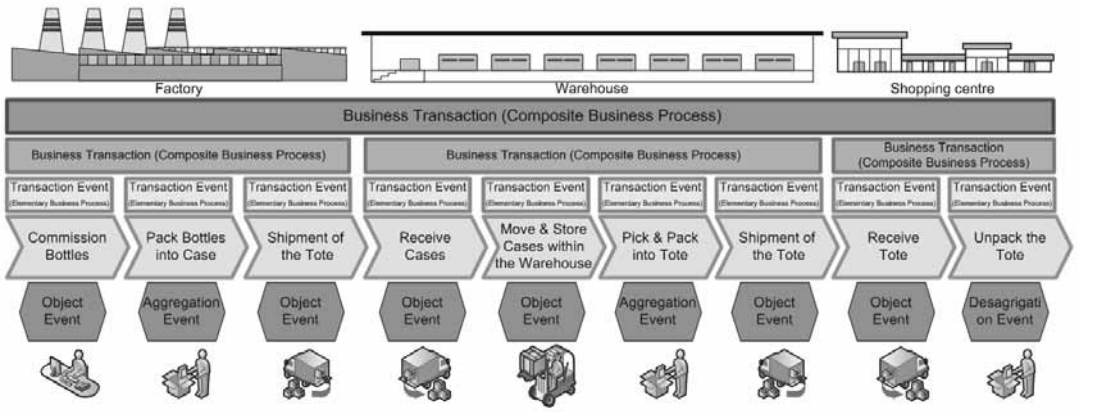

Figure 3.16 EPCIS events for end-to-end processes in the supply chain.

comprise information about a given EPC tag or even a group of tags, which facilitates the implementation of cross-domain interenterprise business processes. GS1 Object Naming Service (ONS) is one of these standards.

3.4.6 Security and Privacy Challenges of the EPCglobal Network

The GS1 architecture provides the means for integrating and deploying large-scale RFID systems. It also specifies a wide range of middleware functionalities that enable enterprise applications to benefit from RFID readings. Nevertheless, the complexity of EPCglobal RFID systems comes with several security and privacy challenges. The latter span all different components and tiers of an RFID system [17].

First, there are security concerns regarding the operation of EPC tags. Even though EPC tags offer some security features, they do not provide access control functions for reading EPCs. Nevertheless, there is access control in the writing functionality of the EPC tags. Another security concern relates to the reading of EPCs that are associated with privacy-sensitive data (e.g., electronic health records). In such cases, there is a need for encrypting the read and transferred data. However, EPC tags offer very limited computational capabilities, which ask for lightweight cryptographic protocols.

At the middleware level, there is a need to provide security functions at the ONS and EPCIS infrastructures of an EPCglobal network. For example, authentication and encryption functionalities are essential in many EPCIS applications. In this direction, basic authentication functionalities found in state-of-the-art web services frameworks can be implemented. However, there are more issues such as the integrity of the ONS and EPCIS service. Integrity is about the correctness and completeness of the information provided by ONS. Such services are susceptible to man-in-the-middle attacks. Specifically, attackers that can gain control of intermediate ONS servers are able to forge the list of EPCIS repositories leading to major security breaches. Another security issue relates to the availability of the ONS and EPCIS service. When EPC networks operate at a global scale, their services are exposed to the internet. Hence, they are susceptible to attacks that can compromise their availability, such as sophisticated distributed denial of service (DDoS) attacks or classic denial of service (DoS) attacks that overwhelm servers based on numerous service requests (e.g., a large number of EPCIS queries). Overall, the operation of a global RFID network creates new points of failure, which could lead to supply chain services disruption.

3.5 Summary

This chapter introduced RFID technology and presented some of the key standards for RFID middleware building blocks. We explained how RFID information can be acquired with the reader protocol, converted to meaningful application-level events based on the EPC-ALE protocol and ultimately transformed to business events based on EPCIS. For the EPCIS protocol, we also presented concrete examples of

events that comprise the business context of RFID readings. Based on these examples, the importance of RFID middleware become evident: applications cannot benefit from tag readings only. Rather, they need to know the business context associated with these readings, including the when, where, and why a tag was read. This is a very important consideration, not only for RFID applications, but for all the different types of IoT applications. It is this contextual knowledge that provides value to consumers and businesses [18].

This chapter complemented Chapters 1 and 2 in terms of the presentation of the most important pervasive sensing devices that are used in the scope of IoT applications. In Chapter 4, we focus on IoT technologies of different nature, namely, technologies that facilitate scalable data processing of IoT data. In this direction, Chapter 4 is devoted to the integration of IoT with the cloud computing paradigm. This integration is a key prerequisite for ensuring that modern IoT applications have access to the computational and storage resources that they need.

References

[1] Wu, D. -L., et al., "A Brief Survey on Current RFID Applications," *2009 International Conference on Machine Learning and Cybernetics*, Hebei, 2009, pp. 2330–2335.

[2] Liu, A. X., L. A. Bailey, and A. H. Krishnamurthy, "RFIDGuard: A Lightweight Privacy and Authentication Protocol for Passive RFID Tags," *Security Comm. Networks*, Vol. 3, 2010, pp. 384–393.

[3] Zavvari, A., "Critical Evaluation of RFID Security Protocols," *International Journal of Information Security and Privacy (IJISP)*, 2012.

[4] Laniel, M., and I. Uysal, "RFID System Optimization for Item-Level Pharmaceutical Serialization," *2013 IEEE International Conference on RFID-Technologies and Applications (RFID-TA)*, Johor Bahru, 2013, pp. 1–6.

[5] GALERIA, https://www.galeria.de/.

[6] European Commission, "Privacy and Data Protection Impact Assessment Framework for RFID Applications," January 12, 2011, https://ec.europa.eu/digital-single-market/en/news/privacy-and-data-protection-impact-assessment-framework-rfid-applications.

[7] Garfinkel, S. L., A. Juels, and R. Pappu, "RFID Privacy: An Overview of Problems and Proposed Solutions," *IEEE Security & Privacy*, Vol. 3, No. 3, 2005, pp. 34–43.

[8] Kefalakis, N., et al., "Middleware Building Blocks for Architecting RFID Systems," *MOBILIGHT*, 2009, pp. 325–336.

[9] Anagnostopoulos, A., J. Soldatos, and S. G. Michalakos, "REFiLL: A Lightweight Programmable Middleware Platform for Cost Effective RFID Application Development," *Pervasive and Mobile Computing*, Vol. 5, No. 1, 2009, pp. 49–63.

[10] Floerkemeier, C., C. Roduner, and M. Lampe, "RFID Application Development with the Accada Middleware Platform," *IEEE Systems Journal*, Vol. 1, No. 2, 2007, pp. 82–94.

[11] Kefalakis, N., et al., "Generating Business Events in an RFID Network," *RFID-TA*, 2011, pp. 223–229.

[12] Byun, J., and D. Kim, "Oliot EPCIS: New EPC Information Service and Challenges Towards the Internet of Things," *2015 IEEE International Conference on RFID (RFID)*, San Diego, CA, 2015, pp. 70–77.

[13] GS1, "The GS1 EPCglobal Architecture Framework," GS1 Version 1.6, April 14, 2014, https://www.gs1.org/sites/default/files/docs/epc/architecture_1_6-framework-20140414.pdf.

[14] "GS1 System Architecture Document - How GS1 Standards Fit Together," Release 9.0, approved February 2020, https://www.gs1.org/docs/architecture/GS1_System_Architecture.pdf.

[15] GS1, "EPC Information Services (EPCIS) Standard," Release 1.2, ratified September 2016, https://www.gs1.org/sites/default/files/docs/epc/EPCIS-Standard-1.2-r-2016-09-29.pdf.

[16] Dimitropoulos, P., and J. Soldatos, "RFID Enabled Fully Automated Warehouse Management: Adding the Business Context," *IJMTM*, Vol. 21, No. 3/4, 2010, pp. 269–288.

[17] Fabian, B., and O. Günther, "Security Challenges of the EPC Global Network," *Commun. ACM*, Vol. 52, No. 7, July 2009, pp. 121–125.

[18] Aggarwal, C. C., and J. Han, "A Survey of RFID Data Processing," in *Managing and Mining Sensor Data*, New York: Springer, 2013, pp. 349–382.

CHAPTER 4

IoT in the Cloud

This chapter is devoted to the integration of IoT with cloud computing. It starts with a discussion of the motivation behind integrating IoT and cloud computing. Accordingly, it introduces various models for using the cloud in the scope of IoT applications. It also presents the edge computing and fog computing models, which are currently the most popular cloud-based deployment models for IoT applications. This chapter ends by presenting tangible examples of cloud-based platforms for IoT, including the platforms of major vendors.

4.1 Introducing Cloud Computing

Cloud computing is an evolutionary step in internet computing, which has completely changed the way that computing resources are accessed. It has also empowered entirely new business models. Cloud computing provides the means for delivering information and communication technologies (ICT) resources as a service (Figure 4.1), including resources such as computing power, storage, computing infrastructure, software applications, business processes, and collaboration services [1]. One of the main characteristics of cloud computing is that it enables access to computing resources regardless of time and the end user's location.

4.1.1 Cloud Computing Properties

From the end user's perspective, cloud computing is like accessing computing resources and software applications over the internet. This has been around for more than 2 decades, under different computing models such as the application service provider (ASP) model. Despite these similarities, the cloud computing paradigm is characterized by a set of properties that differentiate from older models for web-based, on-demand computing such as ASP. These characteristics include:

- *Elasticity:* Elasticity refers to the automated scale-up of cloud computing infrastructures when more computing resources are needed (e.g., during peak loads) and to their automated scale-down when less resources are needed (e.g., during periods with lighter loads). Based on this elasticity property, companies using cloud infrastructure do not have to worry about scaling up

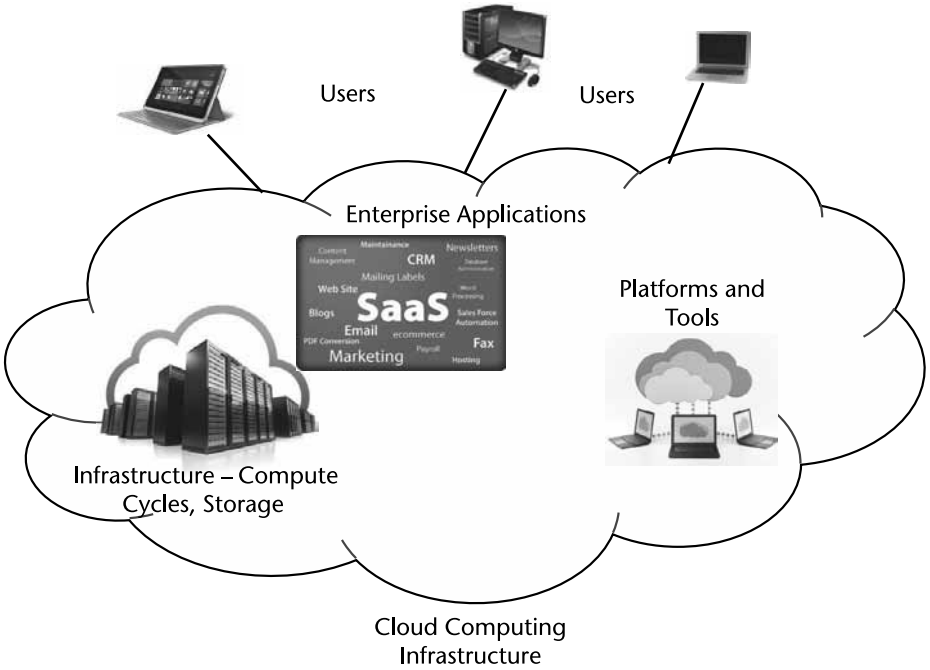

Figure 4.1 Cloud computing infrastructures enable users to access IT resources as a service.

or down the resources that they used. Scale-up and scale-down are automatically handled by the cloud infrastructure based on the needs of the end-user application.

- *Self-service, automated provisioning, and de-provisioning:* In earlier web-based computing models (e.g., internet hosting, ASP), end users had to request from the provider changes in their resource usage plan and to their respective service-level agreement (SLA). With the advent of cloud computing, no lengthy processes for provisioning resources are required. Provisioning and de-provisioning of resources are more automated. They are performed through an online order or even automatically, without human intervention. The latter takes place in cases where the provisioning (or de-provisioning) is foreseen in the SLA between a provider and end users.
- *Application programming interfaces (APIs):* Cloud computing infrastructures come with APIs that enable applications and data sources to communicate with each other. APIs facilitate application integration in the cloud.
- *Pay-as-you-go model:* Cloud computing provides metering and billing of service usage. When using cloud computing services, users pay only for what they use.
- *Performance monitoring and measurement:* Cloud computing infrastructures provide a service management environment. This environment enables the management of physical resources and IT systems.
- *Security:* Cloud computing infrastructures safeguard critical data. This is performed in order to meet customer compliance requirements such as

compliance to security regulations for healthcare services and financial services.

4.1.2 Cloud Computing Stakeholders

Some of the main stakeholders of the cloud computing paradigm are:

- *Cloud vendors and cloud providers:* These provide applications and enable technology, infrastructures, hardware, and integration services.
- *Partners of the cloud providers:* These create cloud services offerings and provide support services to customers (i.e., to the users of the cloud computing services).
- *Business leaders:* These are using or evaluating various types of cloud computing offerings.

These stakeholders are pertinent to the classical cloud computing paradigm. However, they are also relevant for the integration of IoT in the cloud as well.

4.1.3 Cost Models and Business Drivers

In order to illustrate the cost model of cloud computing, we introduce the following terms:

- *Operating expenses (OPEX):* This refers to recurring costs such as costs paid per month or costs charged per user for each service.
- *Capital expenses (CAPEX):* This refers to capital investments (i.e., paying a purchase fee plus yearly maintenance for software within the organization).

The main characteristic of cloud computing is that it enables the use of cloud computing resources-based OPEX (i.e., pay-as-you-go), while at the same time obviating the need for significant upfront investments (i.e., CAPEX).

The rise of cloud computing is driven by two main business needs:

- *Business agility:* In the era of technology acceleration, companies are constantly in need of more computing cycles and increased storage capacity. Hence, they also require flexible IT procurement processes. For example, they seek to implement computationally intensive projects and services without complex IT procurement processes, such as the need to purchase licenses and data center resources.
- *Reduced capital expenses:* Companies are also in need of converting CAPEX to OPEX. This allows them to pay as they go instead of bearing high upfront costs. In this way, they can gain access to the computing resources that they need, whenever they need, without over-provisioning resources. Provisioning resources in advance is risky as it is very common to purchase and pay for resources that will not be ultimately used. Hence, the ability to pay as

you go and to pay for what you use offers distinct advantages when compared to conventional models that rely on significant capital investments. As a prominent example, consider a start-up company that is developing an IoT product or solution. This company may not possess the equity capital needed to establish a computing infrastructure that could accommodate billions of interactions with internet-connected objects. Therefore, it is more practical for this company to employ a cloud computing solution, where upscaling could be gradual as the company grows and gains access to cash flows and revenue streams. The latter streams will finance the OPEX expenses incurred due to the company's access to the needed computing resources.

As explained next, these business drivers are also driving the integration of IoT in the cloud.

4.1.4 Cloud Delivery Models

The three most popular models for delivering cloud services are:

- *Infrastructure as a Service (IaaS):* This is the delivery of storage and computing resources in a pay-as-you-go fashion and in response to requirements for custom business solutions.
- *Platform as a Service (PaaS):* This is the delivery of entire development environments for creating cloud-ready business applications.
- *Software as a Service (SaaS):* This is software applications that are accessed over the cloud in a scalable and pay-as-you-go fashion (e.g., Salesforce.com).

The management of cloud services entails several activities, including:

- *Configuration management:* This deals with the management of the different configurations of the cloud services.
- *Network management:* This includes configuration, security, accounting, and fault management of the network that supports the cloud infrastructure.
- *Capacity planning:* This involves activities towards dimensioning the cloud capacity (e.g., nodes, memory, storage, data center infrastructure) to ensure that the cloud computing infrastructure can accommodate the needs for which it was designed and deployed.
- *Service desk:* This refers to the provision of support services to cloud users, including handling of incidents.
- *Root cause analysis:* This includes activities towards identifying the root cause of problems in the operation of cloud services.
- *Workload management:* This refers to the optimal management of workloads by a given set of cloud resources based on proper actions and policies. Prominent workload management strategies include load balancing across servers and multitenancy.

- *Patch and update management:* This ensures that both the cloud infrastructure and the hosted cloud services remain up-to-date in terms of patches and updates.
- *Asset management:* This refers to the management of cloud resources as assets.

4.1.5 Examples of IaaS and PaaS Clouds

The motivation behind IaaS is that it provides ways for accessing computing resources on demand and at a competitive price. Furthermore, IaaS enables the rapid provision of the computing resources needed for an application without lengthy and tedious procurement processes. Moreover, mainstream IaaS (cloud) providers offer higher security than most of the on-premise data centers. Also, IaaS provides on-demand and flexible access to computing resources, which facilitates the business agility of companies that use IaaS.

The most famous example of IaaS is Amazon's Elastic Compute Cloud (EC2), which is the highest-profile IaaS operation. EC2 uses Xen virtualization [2] to create and manage virtual machines. Xen is a popular, proven, open-source hypervisor. Amazon allows the creation of virtual servers in one of three sizes: small, large, or extra-large.

EC2 provides persistent storage for those who want it, in the form of elastic block storage (EBS). Users can set up and manage storage volumes. It is also possible to connect these EBSs to servers, so the data are attached to the server instances.

Amazon EC2 pricing is based on one of the following models:

- *Primary charges:* These include hourly charges per virtual machine and/or data transfer charges.
- *Hourly charges:* These are counted from the moment that a virtual machine (VM) is created to the time that it is taken down. In the case of this hourly mode, the charge applies independently of whether the resources are fully used or lying idle.
- *Data transfer charges:* These involve charges for data input and output (I/O) rather than data retained in the cloud data center. This leads, for example, to increased rates for running operating systems (OS) and only small charges for data transfer between instances.

The PaaS model provides a deeper set of capabilities when compared to IaaS. PaaS enables access to a cloud computing platform that includes a set of development, middleware, and deployment capabilities [3]. A PaaS environment builds an ecosystem around it. Most PaaS vendors try to create and support a deep ecosystem of partners who all commit to this environment for the future.

Three types of PaaS platforms are distinguished, including:

- *Integrated application development platforms:* Examples are Google App Engine and Microsoft Azure. They provide a workflow engine, development

tools, a testing environment, an ability to integrate databases, and various third-party tools and services.

- *Enterprise application platforms:* Examples are Salesforce.com and Force.com. They are like the integrated application development platforms, yet they also provide access to business software and enterprise application (e.g., CRM, ERP).
- *Enabling technologies and support services platform:* These are platforms providing specialized capabilities, such as cloud-based monitoring and configuration, testing and deployment, and social networking services.

4.2 IoT and Cloud Convergence

4.2.1 IoT and Cloud Integration: The Motivation

The convergence of IoT with the cloud is motivated by the need to enable IoT applications to leverage the benefits of the cloud, including scalability, capacity, and quality of service (QoS). Likewise, IoT vendors integrate their applications to the cloud in order to benefit from flexibility in provisioning and accessing resources [4]. The following business scenarios illustrate the business need for IoT and cloud convergence:

- *Supporting IoT start-ups with limited equity capital:* Most IoT start-ups lack the equity capital needed to undertake significant investments in computing and storage infrastructure. Cloud computing provides them with the flexibility in accessing the resources that they need without significant upfront investments. Small medium enterprises (SMEs) can pay for the cloud resources that they use. In this way, they can postpone costly investments for later times, when they will have grown and will be able to afford significant pay-as-you-go costs. Overall, cloud computing enables IoT start-ups to pay as they grow.
- *Access to computational resources for novel AI products and services:* Nowadays, many IoT enterprises innovate based on machine learning and AI. AI and machine learning applications tend to be computationally and storage-intensive, which asks for access to large amounts of computation and storage resources. Cloud computing offers a compelling solution for on-demand access to the needed resources for training and executing AI/machine learning algorithms. Based on cloud computing, IoT companies do not need to purchase resources that will be idle and underutilized.
- *Accessing IoT devices and tools in the cloud:* Companies developing IoT products and services can nowadays access IoT development and deployment tools in the cloud. This obviates the need for these companies to install many different tools locally. Rather, they can access all the IoT development and deployment resources that they need from a single point and regardless of time and the location of their employees. This provides IoT innovators with flexibility when developing their products and services.

Despite these benefits, the integration of IoT with the cloud is not straightforward. Specifically, the integration must deal with the conflicting properties and diverse nature of IoT and cloud computing resources [5, 6]. IoT resources are location-specific, resource-constrained, and associated with quite high development and deployment costs. Moreover, IoT devices and resources are generally inflexible in terms of resource access and availability. However, computing resources in traditional cloud computing are location-independent and generally inexpensive in their deployment. Furthermore, the cloud provides rapid elasticity on computing resources and flexibility regarding their provisioning and use.

4.2.2 History of IoT and Cloud Convergence

Given the challenges of IoT and cloud integration, early efforts have focused on streaming of sensors and WSN data in a cloud infrastructure [7]. Most of the early efforts were carried out in the scope of research initiatives. In 2007, we witnessed the first commercial efforts towards integrating data and services from IoT devices within cloud computing infrastructures. A wave of IoT-cloud platforms emerged accordingly. Some of the enterprise efforts led to platforms such as Xively (https://xively.com), ThingWorx (www.thingworx.com), ThingSpeak (https://thingspeak.com), sensor-cloud (www.sensor-cloud.com), and RealTime.io (https://realtime.io/). Since 2012, open-source platforms have also emerged such as the OpenIoT project (https://github.com/OpenIotOrg/openiot) [4]. These platforms enable the integration of IoT data sources within a cloud and provide a wide range of tools and APIs for developing applications over these data. However, each one of these IoT or cloud platforms comes with its specific added-value functionalities, which differentiate itself from competitors. In recent years, the vast majority of nontrivial IoT applications have been cloud-based.

4.2.3 IoT and Cloud Platform Characteristics

As discussed, the challenges of IoT and cloud integration stem from the different nature of IoT and cloud computing resources. These challenges can be classified in the following main categories:

- *The data quality of various sources:* In IoT, the accuracy of each data point can be debated, depending on the reliability of the IoT device and the availability of the respective IoT data. Moreover, the time of measurement is important along with the trustworthiness of the data source.
- *The lack of interoperability:* As different data arrive in the cloud, they feature different semantics, such as different reference systems and units (e.g., nGy/s, mSv/h, Sv/h, Bq/kg, cpm). This makes it very difficult to use, integrate, or even compare data arriving in the cloud from different IoT devices, IoT platforms, and other IoT data sources.
- *The mixing of data sources:* Sensor and IoT information in the cloud is not always directly associated with some physical device. It is also possible to have virtual sensors (e.g., data scraping from web pages) in addition to real

physical sensors. The cloud infrastructure for IoT applications shall be able to cope with data from both physical and virtual sensors.

To alleviate these challenges, IoT-cloud platforms implement the following solutions and features:

- *Management of heterogeneity:* This is done through the specification and implementation of abstraction layers (e.g., virtual sensors in the WSN).
- *Adaptation of popular solution deployment models to IoT:* This includes the IaaS, SaaS, and PaaS models. Supporting such models for IoT resources requires some adaptation to the peculiarity of IoT devices and computing resources.
- *Handling large volumes of high-velocity data in the cloud:* This is done through integrating big data techniques in the cloud, including streaming middleware frameworks. IoT data analysis and big data integration in the cloud are discussed later in the book.
- *Providing support for application development:* This is done through integrated environments for cloud-based IoT applications that provide APIs and mashups for service developments.

4.2.4 Delivery Models for IoT Services in the Cloud

As presented earlier, popular cloud service delivery models can be adapted in IoT cases as:

- *IaaS for IoT:* IaaS services for IoT access IoT infrastructures in the cloud (e.g., they access sensors and actuators). The business model of such cloud services is based on a data or IoT device provider model (i.e., the user of the service pays for accessing the data of the IoT device or, more generally, for using the IoT device). IaaS cloud services for IoT provide access control to IoT resources.
- *PaaS for IoT:* This is nowadays the most widespread model for IoT services delivery in the cloud. It is the model followed by most of the already presented public IoT clouds (e.g., xively.com). The PaaS model is based on access to data and services of physical or virtual devices. It also offers complete development environments (i.e., tools and utilities) for implementing IoT applications.
- *SaaS for IoT:* This refers to IoT software applications that are deployed in the cloud. These are typically applications developed by the application development environments provided by PaaS for IoT. IoT software applications are developed for various application domains such as smart buildings, smart homes, smart cities, and smart health. SaaS IoT services are typically based on pay-as-you-go (i.e., utility-based) business models.

Not all IoT services can be deployed within a cloud. Specifically, IoT services can be also classified as:

- *On-device or on-resource services:* These are low-level IoT services, which are location-dependent and cannot be deployed on the cloud. They are services that are based on the deployment of a given device on a certain location.
- *Entity services:* These are services based on the composition of multiple low-level services. They are typically able to provide information about an entity and can be deployed in the cloud.
- *Complex value-added services:* These services are based on the composition of entity services with physical world services. They are also cloud deployable.

4.2.5 IoT and Everything as a Service

In addition to supporting and providing IoT services based on the popular cloud models (e.g., IaaS, SaaS, PaaS), IoT and cloud integration also enable additional models. One of these models is the sensing-as-a-service model [23], which enables the dynamic specification of sensing services based on the processing of data from various available sensors. The sensing-as-a-service paradigm entails the on-demand selection and establishment of IoT sensing services given a set of available sensors. It hinges on the location-dependent nature of IoT resources and is based on the dynamic selection of the sensors and devices used to deliver the service. It is a very common model for IoT and utility computing convergence, where a sensing service is created on demand and is priced on a pay-as-you-go-fashion. In many cases, it can be considered as a special case of a SaaS for IoT applications.

IoT services can be orchestrated to deliver entire IoT-based workflows as a service. These workflows may include IoT services beyond sensing, such as actuation services or IoT data processing services as a service. Hence, cloud-based IoT models can be combined to support the notion of Everything as a Service (i.e., the orchestration of IoT services in the cloud without restrictions in the number and type of devices involved). Future applications such as IoT-based industrial automation are likely to incorporate many cloud-based IoT workflows in line with the Everything-as-a-Service concept.

4.3 Edge Computing

4.3.1 Limitations and Problems for IoT-Cloud Integration

The integration of data and services from IoT devices in the cloud is associated with several limitations including:

- *Waste of bandwidth:* Given that not all IoT data need to be stored in cloud, this leads to a waste of bandwidth, especially in cases of large-scale ap-

plications, where data from multiple sensors are continually or frequently integrated in the cloud.

- *Network latency:* Given that interactions of IoT devices with the cloud are not network-efficient, this can be a problem for real-time applications.
- *Inefficient use of storage:* Information with limited (or even zero) business value is likely to be stored in the cloud, consider, for example, sensor data that does not change frequently (e.g., temperature information). Continually storing such data in the cloud leads to wasted storage resources.
- *Limited flexibility to address privacy and data protection:* All data are stored to the cloud without any easy way to "isolate" private data (e.g., a citizen's personal data).
- *Cloud data are "away" from users:* This makes the cloud a rather bad choice for supporting applications that involve mobility and processing of data close to the field. In most cases, IoT devices access the cloud through a metropolitan area network (MAN) or a wide array network (WAN), rather through a local area network (LAN). Hence, cloud processing incurs high latency and cost.

These challenges are intense for specific types of applications, such as real-time applications that involve operations close to the field and applications involving mobility and frequent handovers of users' devices from one provider to another.

4.3.2 The Edge Computing Distributed Paradigm

To alleviate the above outlined limitations, the edge computing paradigm has been introduced [8]. The basic principle of edge computing is to move IoT data processing and actuation to the edge of the network. Specifically, edge computing refers to a distributed computing paradigm that brings computational and storage resources closer to the location where such resources are needed (i.e., typically closer to the field). The main rationale behind this distributed paradigm is to reduce latency, to improve response times, and to economize on the network bandwidth used. In practice, with edge computing, the number of processes that run in the cloud is reduced, as some processes are moved closer to the field (i.e., to the user's device), to an IoT device, or even to an edge server. Hence, computational power is near end users and their IoT devices such as sensor gateways, smartphones, and other technologies. A characteristic example of an edge computing deployment involves the introduction of a layer of gateways (i.e., edge nodes) between the cloud and the IoT devices. Edge computing implementations may involve a variety of node types that enable different deployment paradigms. The different node types include, for example:

- Embedded controllers or IoT devices with some processing capability;
- Computers;
- Entire clusters or small-scale data centers (local clouds).

Edge computing enables a new, massively decentralized computing paradigm, which is very well-suited for IoT applications [9]. Figure 4.2 illustrates the decentralization of a cloud architecture to an edge computing one, through splitting the application logic across cloud and edge nodes. This can lead to increased scalability and better use of available bandwidth. Overall, the benefits of the edge computing approach include:

- Reduced latency for real-time applications;
- Efficient use of bandwidth and storage resources;
- Improved scalability for very large applications;
- Reduction in costs and energy consumption;
- Better privacy control.

The typical functionalities of IoT applications that are implemented and deployed in edge nodes include [10]:

- *Data filtering and reduction:* This is as a means of optimizing bandwidth and storage. Filtering functionalities get rid of meaningless data and forward to the cloud only the part of the data that is relevant to the target application logic.
- *Reaction (i.e., actuation and control):* This is as a means of performing actuation and control operations close to the IoT devices (i.e., close to the field). This facilitates real-time operations.
- *Caching for fast access and (re)use of data:* This is as a means of accelerating data access and processing in the scope of real-time applications.

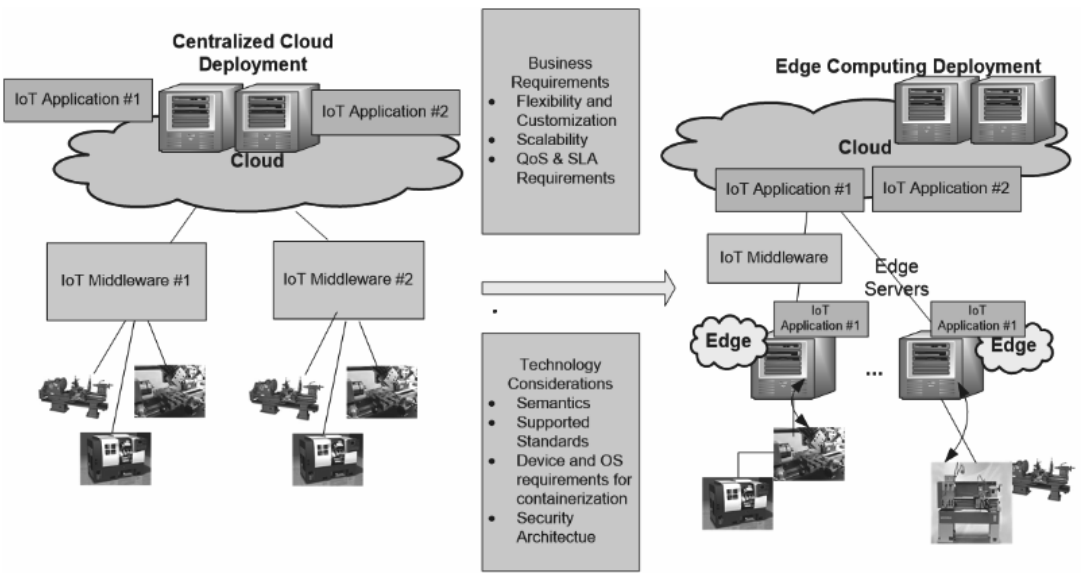

Figure 4.2 Edge computing concept.

Furthermore, edge nodes are likely to perform data analytics in order to support real-time applications that require analytics close to the field [11].

Edge nodes and edge devices are deployed in conjunction with a cloud computing deployment, which is the reason why edge computing deployment is commonly characterized as an edge-cloud deployment.

4.3.3 Fog Computing Versus Edge Computing

Fog computing and edge computing are effectively the same thing, which explains why the fog computing term is, in some cases, used interchangeably with edge computing. However, there is a slight difference between the two concepts:

- In the scope of fog computing, data are processed within a computing node or IoT gateway that resides within the LAN of the deployment [12]. Hence, fog computing involves transferring data to fog nodes. In fog computing, intelligence is pushed to a fog node or an IoT gateway.
- In the scope of edge computing, data are processed on edge device. Hence, data are not transferred to any other device. Rather, the intelligence and processing paradigm is provided directly from edge devices such as automation controllers of programmable logic controllers (PLC).

Thus, there is a difference on where the intelligence is placed and where the processing is done. Nevertheless, both paradigms have the same goals and effects (i.e., support for low-latency operations and optimization of network bandwidth).

In a typical fog computing, deployment signals and data from the IoT devices (i.e., things and internet-connected objects such as a PLC) are sent to some server or protocol gateway, which undertakes to convert the data from the low-level protocols (e.g., PLC I/O protocol) to some protocol understandable by IoT systems and networks such as message queuing telemetry transport (MQTT) or Hypertext Transfer Protocol (HTTP). Accordingly, the data are sent to some IoT gateway on the LAN that collects the data and processes it at higher layers. The processing may entail filtering, analysis, and processing of the data from the devices, as well as storage of the data for later transmission to the cloud. Overall, a fog computing architecture comprises various layers of data transfer and processing, while carrying out various data conversions.

In an edge computing architecture, the communication pipeline is simpler. Specifically, data from the physical internet-connected devices are directly connected to a control system at the edge of the network. This control system is conveniently characterized as an edge device and host processing programs allow it to directly collect, analyze, and process data from the physical devices. Moreover, the edge device can execute actuation and control logic by directly interacting with the physical devices (i.e., things, internet-connected objects). Such edge devices can therefore implement edge intelligence (i.e., they allow pushing the intelligence to the network edge where physical assets and things reside at the first place). Overall, edge computing architectures are more simplified than fog computing architectures, as

they completely streamline communications. Moreover, they have fewer potential points of failure than fog computing architectures.

Figure 4.3 illustrates the concept of the edge, fog, and cloud computing parts of an IoT system. Passive IoT devices without computational capabilities connect typically to fog nodes in order to leverage the benefits of the fog computing paradigm. However, edge devices have computational capabilities that enable data processing close to the field, without a need to integrate them with a fog node. Both fog nodes and edge nodes provide their data to their cloud, where datasets from different devices can be aggregated and processed in an integrated way. The edge, fog, and cloud computing parts of an IoT deployment are usually presented in a pyramid form as in Figure 4.3. This is because the number of field devices is much larger than the number of edge devices, which are accordingly more than the number of available fog devices (e.g., gateways).

Despite their architectural differences, both edge and fog computing systems alleviate the limitations of cloud computing for IoT applications. This is why popular reference architectures for the IoT, such as the Reference Architecture of the Industrial Internet Consortium (IIRA) [13] and the Reference Architecture that has been produced by the OpenFog Consortium[1], indicate edge computing and fog computing as the primary ways for architecting and implementing IoT systems.

4.3.4 Edge Computing Applications

Popular examples of IoT applications that are suitable for edge computing are:

- *Mobile applications:* Examples are applications involving fast-moving objects (e.g., connected vehicles, autonomous cars, connected rail) and/or ap-

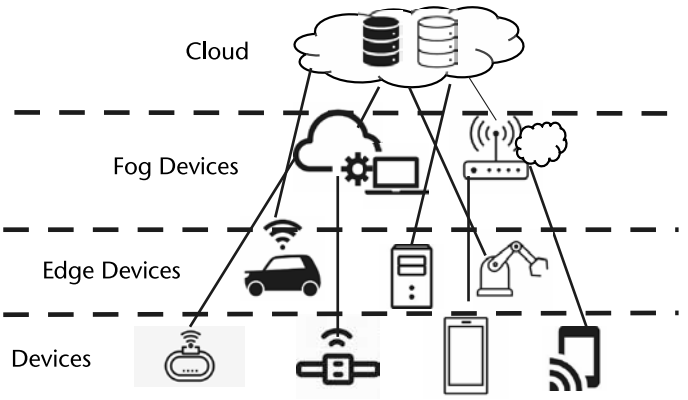

Figure 4.3 Overview of the fog, edge, and cloud parts of an IoT system.

1. In 2019, OpenFog Consortium members were continuing their standardization work in the scope of the Industrial Internet Consortium.

plications requiring the interfacing of moving devices or objects to local resources (computing, storage).

- *Large-scale distributed control systems:* Examples are smart grid, connected rail, and smart traffic light systems. Such applications combine geo-distributed properties and real-time applications. Edge computing enables them to deal with scalability and latency issues.
- *Distributed multi-user applications with privacy implications:* These include applications in need of fine-grained privacy control (such as processing of personal data). In the scope of such applications, the edge computing paradigm provides decentralization of the storage and management of private data to the various edge servers. Hence, these applications provide no risks in transferring, aggregating, and processing all private datasets at the centralized cloud computing infrastructure. In this way, users have better and isolated control over their private data.

4.3.5 Multi-Access Edge Computing (MEC)

4.3.5.1 Overview

A special class of edge computing deployments is those involving mobile objects or devices. Relevant edge computing deployments are characterized as mobile-edge computing deployments and involve [14]:

- *On-premise isolation:* They are isolated from the rest of the IoT network to enable machine-to-machine (M2M) applications with increased security. This renders M2M applications less prone to errors at other points of the network.
- *Proximity for low-latency processing:* This happens through enabling access to mobile devices and running services close to them. This leads to reduced latency, bandwidth savings, and optimal user experience.
- *Location and context awareness:* This enables location-based services. This supports new business services that utilize network context and locations (e.g., users' context, points of interest, and events).

As evident from these applications and use cases, MEC introduces a novel network architecture concept, which complements conventional cloud computing with a pool of IT services at the edge of the network. MEC's core concept lies in the execution of data processing and application control tasks closer to the cellular customer as a means of reducing network congestion and optimizing application performance. MEC has been introduced and is standardized by the European Telecommunications Standards Institute (ETSI) [15]. From an implementation perspective, MEC is realized based on the deployment of edge nodes (such as cellular base stations) between the cloud and the customer. MEC combines both IT and telecommunications technologies, which enables flexible and rapid deployment of new applications and services for customers [16].

4.3.5.2 The MEC Ecosystem

MEC specifies and enables a new ecosystem and value chain for mobile networked services. In the scope of the MEC ecosystem, mobile operators open their radio access network (RAN) edge to authorized third parties such as application developers and content providers. By accessing the RANs of cellular operators directly, third parties can develop, deploy, and offer a wide range of novel applications that are characterized by ultra-low latency and high bandwidth availability. Such novel applications are typically offered to mobile subscribers, enterprises, and vertical segments. The MEC ecosystem provides business opportunities to all stakeholders:

- *Mobile network operators (MNOs):* These are offered with opportunities for deploying innovative services for their consumer and enterprise business segments, including services that could differentiate their service portfolio and set them apart from competitors. Based on MEC, MNOs can improve their bottom lines through reducing the volume of signaling in their core networks, charging content providers and application developers for the resources that they use, and generating new revenue streams from innovative services.
- *Application developers and content providers:* These can develop and deploy novel services over the MNOs' infrastructures, notably through direct access to the RANs. They are provided with access to IT resources at the edge of the network, which enables them to provide a host of IoT and big data applications.
- *End users (e.g., mobile subscribers):* These are provided with access to novel applications that are enabled by MEC's key features (e.g., latency, privacy management, effective bandwidth). Applications such as predictive maintenance and field service inspections based on drones were hardly possible before the advent of edge computing. Likewise, end users benefit from improved user experience (e.g., in augmented reality (AR) applications) and cost-effective applications' operation.

4.3.5.3 MEC Platform Architecture

According to the ETSI MEC standard, a MEC platform comprises a hosting infrastructure and an application's platform that supports the provision and execution of various applications. The hosting infrastructure consists of hardware resources and a virtualization layer. Its implementation is abstracted from the applications that are hosted on the platform, which means that application developers need not be aware of the low-level details of the hosting infrastructure such as its hardware components.

The MEC application platform comprises a set of middleware services, which are offered to the applications that are deployed and hosted on the MEC services. These services include:

- *Infrastructure services:* These include a range of communication services and a service registry structured according to a service-oriented architecture (SOA) approach. These services facilitate the communication between MEC applications and application platform services through proper APIs. However, the service registry provides visibility on the services of the MEC server, as means of enabling deployment flexibility in dynamic environments.
- *Radio network information services (RNIS):* These deliver information about users and cells from the radio network. For instance, RNIS provides information about the activation of user equipment (UE) on a specific mobile network element, as well as information about the location of the subscriber, cell load, and throughput guidance.
- *Traffic offload function (TOF):* This prioritizes traffic and routes the selected, policy-based, user-data stream to and from applications that are authorized to receive the data. TOF operates in two modes: (1) a pass-through mode where (uplink and/or downlink) traffic is passed to an application that can monitor, modify, or shape it and then send it back to the original packet data network (PDN) connection; and (2) an end-point mode where the traffic is terminated by the application that acts as a server. These two modes enable different applications and use cases.

The platform provides an application management interface, which enables operators to configure the platform and to manage the life cycle and operability of the MEC applications. Beyond application platform management functionalities, the MEC specifies the possibility of implementing application-specific management systems that are tailored to the business logic of the respective applications.

4.3.5.4 MEC APIs

Open APIs are among the key enablers of the MEC ecosystem as they facilitate the development of novel applications by application developers. Furthermore, they ease the adaptation of existing services and applications to the MEC environment. ETSI has therefore released a first set of MEC APIs, which are described in the following specifications:

- *GS MEC 009—General Principles of Mobile Edge Service APIs:* This outlines the principles that drive the definition of RESTful multi-access edge service APIs, including naming conventions and relevant API patterns.
- *GS MEC 010-2—Mobile Edge Management, Part 2:* This focuses on application life-cycle management at the edge of the network, including interfaces, reference points, and the rules.
- *GS MEC 011—Mobile Edge Platform Application Enablement:* This outlines how applications communicate with the edge with emphasis on the information flows between the application and the platform, as well as on relevant data models.

- *GS MEC 012—Radio Network Information API:* This focuses on radio network information to mobile edge applications and mobile edge platforms.
- *GS MEC 013—Location API:* This provides the means for tracking device locations and building location-based services.

4.3.5.5 MEC on the Road to 5G

MEC's functionalities and benefits are perfectly aligned to some of the main promises of the fifth generation of mobile communications (5G), which is the emerging telecommunication platform that will support future IoT applications [17]. In particular:

- MEC enables significantly faster access to and delivery of data, which is in line with 5G's promise to increase data speeds up to 10 Gbps.
- MEC boosts applications that require ultra-low latency, including some of the applications that shape the development of 5G such as industrial applications (Industry 4.0), driverless cars, and tactile internet applications.
- MEC eases the connectivity of devices and smart objects, which is in line with 5G's target to boost the deployment of internet-connected devices as part of IoT applications.

Beyond this high-level alignment of main goals and benefits, MEC is already employing some of the tools and techniques that will be used by 5G as well. For example, MEC supports multiple RANs and caters for the connectivity of different devices, while at the same time defining APIs that could enable SOA and microservice deployments, which will be at the very core of 5G.

Based on the above alignment, MEC can realistically deliver today some of 5G's anticipated functionalities. While 5G will be providing a superset of MEC's functionalities, it is still not widely available. Hence, MEC is nowadays a very compelling option for supporting the requirements of emerging IoT and Industry 4.0 applications, which can be a catalyst for the digital transformation of industrial organizations. 5G is likely to support MEC as an enabler for its ultra-reliable low-latency communication (URLLC) services targeting applications that require submillisecond latency with error rates lower than 1 packet loss in 10^5 packets. This will ensure 5G compatibility for the wave of MEC applications that will be deployed until 5G becomes commercially available. The 5G of mobile communications and its importance for emerging IoT deployments are presented in Chapter 9.

4.3.6 Public Cloud Infrastructures for IoT

Public IoT clouds are publicly accessible infrastructures that deliver cloud services for IoT, based on one or more of the above models. There are many public IoT clouds in the IoT market, serving different market segments and end-user needs. Some characteristic examples follow.

4.3.6.1 Xively

One of the most prominent examples of public IoT platforms is Xively, a commercial PaaS for IoT. Xively is one of the first entrants in the IoT-cloud space, which evolved from the famous Pachube, one of the first web-based and cloud-based IoT platforms that emerged more than 10 years ago. Xively supports hundreds of platforms, millions of gateways, and billions of smart devices in the cloud, based on a range of scalable, comprehensive, and secure infrastructure services. It also provides online development tools and a development center to enable developers to write applications that integrate multiple devices and IoT applications. The development and integration of such applications enable a best-of-breed approach. The main features of the platform include:

- Fast and infinitely scalable connectivity (in 2016, it supported and processed over 86 billion messages per day);
- Management functionalities including provisioning, monitoring, updating, and user management;
- Engagement features such as actionable insights based on big data processing and integration with other enterprise information systems.

The main components of the xively.com platform include:

- *Devices:* Enterprise systems and connected objects are integrated to the cloud based on the Xively API. The latter is very lightweight and easy to use, since it supports REST, socket and MQTT programming.
- *Message bus:* This enables real-time management and routing of messages from the various connected systems and devices. This means that Xively provides a message-oriented middleware that enables the handling of messages from the connected systems and devices in a low-overhead asynchronous way, which is appropriate for supporting real-time applications.
- *Object directory:* This keeps track of all objects and devices that are connected to the platform. The directory enables authorized parties to search for objects and their properties, based on criteria such as the type and location of objects.
- *Data management services:* These provide the means for storing large amounts of data (big data) in the cloud, including historic data from the various connected devices.
- *Business services:* These enable the activation, provisioning and management of the objects and devices comprising the Xively public cloud platform.

4.3.6.2 ThingWorx Platform by PCT

ThingWorx is another popular IoT-cloud platform that is provided by PCT to enterprises that aim at building Industrial IoT applications. It is an enterprise-ready technology platform that enables innovators to rapidly develop and deploy smart,

connected solutions for IoT. It claims an ability to enable reduction in application development effort by 10 times. This effort reduction is based on:

- *Model-based development:* This enables applications developers to leverage IoT application models.
- *Search-based intelligence:* This facilitates the discovery and use of things and services.
- *Ecosystem of ThingWorx ready extensions:* These include libraries that boost application development.

The main technical and technological pillars of the ThingWorx platform include:

- *ThingWorx foundation:* This is a set of core libraries, which connects to all the ThingWorx components. It enables a simplified, seamless approach for developers to create comprehensive IoT solutions.
- *ThingWorx utilities:* This is a comprehensive set of tools for business users to define, monitor, manage, and optimize the performance of their connected products.
- *ThingWorx analytics:* This is a framework (including libraries and APIs), which enables IoT developers to quickly add real-time pattern and anomaly detection, as well as predictive analytics to their solutions. The framework provides the means for simulating IoT scenarios.
- *Vuforia Suite:* This is a set of powerful technologies that operate over the platform to facilitate AR development. It emphasizes scalability and integration with IoT solutions, while being one of the distinguishing characteristics of the ThingWorx platform, when compared to other platforms.

4.3.6.3 ThingSpeak

ThingSpeak is another example of IoT-cloud platform, which has been around for several years. Its main features and functionalities include real-time data collection and storage, MATLAB analytics and visualizations, management of alerts, scheduling of events, and device communication. These features are accessible via an OpenAPI, which provides support for geolocation data as well.

Among the main distinguishing characteristics of this platform is that it provides readily available interfaces and integration with a wide array of IoT-related developments and ecosystems, including integration with Arduino, Particle Photon and Core, Raspberry Pi, Twitter, Twilio, and MATLAB. It also provides libraries, APIs, and support for the development of mobile and web apps.

4.3.6.4 Microsoft Azure IoT

In addition to the platforms presented previously, all major IT vendors have been recently developing and releasing cloud-based IoT platforms. A key element of the

IoT strategy of large vendors is that they capitalize on their existing ecosystems in terms of cloud infrastructures, developers, and partners. Hence, they are creating and promoting their own cloud-based IoT platforms, which enable their existing developers' base and users' base to take advantage of IoT technology in the cloud.

Microsoft is an example of a giant IT vendor that has entered the IoT space via a cloud platform, namely the Microsoft Azure IoT, which is built on top of Microsoft's popular Azure platform. Microsoft Azure provides a PaaS infrastructure for IoT solutions. It is primarily destined for Microsoft-based public cloud IoT services, but it can enable private and hybrid cloud implementations as well.

The Azure IoT platform is based on Microsoft Azure IoT reference architecture, which defines a blueprint for architecting and implementing IoT solutions based on Microsoft's technologies. Solutions adhering to the reference architecture enable the flow of information between intermittent or continuously connected devices or even line-of-business assets and cloud-based backend systems for the purpose of analysis, control, and business process integration. The architecture is designed to address large-scale IoT environments with devices from industrial serial production with tens of thousands of units and/or industrial machinery emitting significant amounts of data.

Microsoft's IoT reference architecture is driven by the following principles:

- *Heterogeneity:* It is designed to accommodate various scenarios, environments, devices, processing patterns, and standards. Hence, it is suitable for IoT's hardware and software heterogeneity.
- *Security:* It ensures security and privacy measures across all relevant areas (i.e., device and user identity, authentication and authorization, data protection, data attestation). Moreover, it addresses both data at rest and data in motion.
- *Hyper-scale deployments:* The architecture addresses scalable deployments supporting millions of connected devices in the cloud. It also supports the transition (scale-out) of small projects to hyper-scale dimensions (e.g., based on pay-as-you-grow principles).
- *Flexibility:* The Microsoft architecture is driven by a principle of composability based on extension points and the usage of various first-party or third-party technologies. It is a high-scale, event-driven architecture with brokered communication, which enables loosely coupled composition of services and processing modules.

This Microsoft Azure IoT reference architecture includes three main layers (planes):

- *Device connectivity plane:* This plane includes the IoT devices, which are connected to the cloud platform. IoT devices can be IP-enabled or based on other protocols. Moreover, devices can be either directly connected to the cloud platform (e.g., this is the case of the IP devices) or through an IoT gateway (e.g., edge IoT devices). Low-power devices are integrated to

the IoT-cloud platform through the IoT gateway, since they do not typically possess the computationally capability needed to run an IoT client and/or an IoT protocol.

- *Data processing, analytics, and management plane:* This plane comprises several modules, which are used to store data and metadata about the various IoT devices and data streams. It also includes modules for the processing of IoT streams, including IoT analytics and machine learning modules. One of the main modules of this plane is the "identify and registry store," which is a database of things that includes information about the state of the various devices.
- *Presentation and business connectivity plane:* This plane comprises the modules for the presentation and visualization of an IoT solution, as well as for the integration of the IoT part of a solution with other business systems. The presentation is supported on both mobile and desktop devices based on an appropriate UI. As far as the integration with enterprise systems is concerned, the architecture specifies various connectors and gateways. The latter are used as connection bridges to business information systems (such as enterprise resource planning (ERP) and customer relationship management (CRM) systems).

This reference architecture outlines the basic components of a Microsoft Azure-compliant IoT solution. However, it is also indicative of a structure of a complete integrated IoT solution, which integrates information and services from IoT devices with other business systems, while presenting IoT information to some mobile or desktop app. Similar reference architectures were briefly discussed in Chapter 1.

4.3.6.5 Amazon AWS IoT

Amazon is one more example of a giant vendor that provides IoT services in its cloud, as part of its Amazon AWS IoT platform. AWS IoT provides secure, bidirectional communication between internet-connected things and the AWS cloud. It provides support for sensors, actuators, embedded devices, smart appliances, and other IoT components. It enables collection of telemetry data from multiple devices, as well as data storage and analysis of IoT data. Moreover, it enables users to control these devices from their phones or tablets.

The Amazon AWS IoT platform specifies its own architecture for IoT solutions. This architecture specifies the components of an AWS IoT solution, along with the structuring principles among them. AWS IoT makes provisions for the integration of IoT applications to other services of the popular Amazon AWS cloud platform such as the Amazon S3 data storage services, as well as the Amazon DynamoDB no-SQL cloud database [18]. Furthermore, the Amazon AWS IoT architecture makes provisions for the integration of "things" to these services through an appropriate software development kit (SDK) (Thing SDK) and in a message-oriented way. To this end, the Amazon AWS IoT platform includes a message broker infrastructure. Similar to the Microsoft IoT Azure architecture, the Amazon AWS IoT architecture includes a registry of things. It also comprises a rules engine, which

provides business rules for integrating IoT services and data to existing services of the Amazon AWS IoT infrastructure.

The AWS IoT platform provides the following main interfaces for developers and deployers:

- *AWS Command Line Interface (AWS CLI):* This runs commands for AWS IoT on Windows, OS X, and Linux. The API supports creation and management of things, certificates, rules, and policies.
- *AWS IoT API:* This allows one to build IoT applications using HTTP or HTTPS requests. It enables one to programmatically create and manage things, certificates, rules, and policies.
- *AWS SDKs:* This includes a set of language-specific APIs and wraps the HTTP/HTTPS API. It allows one to program an application in any of the supported languages.
- *AWS IoT Device SDKs:* This can be used to build applications that run on devices. They support sending messages to and receiving messages from AWS IoT.

4.3.6.6 Sentilo Platform

An example of a cloud platform for IoT beyond platforms of large IT vendors is the Sentilo Platform. It is an open-source sensor and actuator platform designed to fit in the smart city architecture of any city. It emphasizes openness and easy interoperability. It is built, used, and supported by an active and diverse community of cities and companies. Sentilo is designed as a cross-platform with the objective of sharing information between heterogeneous systems and easily integrating legacy applications. Sentilo's main functionalities include:

- A front-end for message processing, with a simple REST API;
- An administration console for configuring the system and managing the catalog;
- A memory database aimed to accomplish high-performance rates;
- A non-SQL database, in order to get a more flexible and scalable system;
- A universal viewer, provided as a public demo, that can be used as a start point for specific business visualizers;
- A basic statistics module that records and displays basic platform performance indicators.

Sentilo is an extensible component-based architecture, which makes it possible to enhance the platform functionality without modifying the core system.

The Sentilo platform architecture includes three layers:

- *Layer of IoT data and services providers:* This provides the means for accessing data and services from IoT devices. Access is other directly or

through some gateway (e.g., edge device), which provides relevant protocol adaptation.

- *Data storage and processing layer:* This provides storage functionalities along with a range of data processing agents. At this layer, publish-subscribe functionalities to the various devices are also implemented. There is also a catalog of devices and things, similarly to the case of the "things registry" in the previously described platforms.
- *Application development and deployment layer:* Here the services of the data processing layer are used and integrated in the scope of IoT apps.

The three layers communicate through mainstream protocols such as HTTP and based on relevant HTTP-based APIs.

4.3.6.7 The OpenIoT Platform

The OpenIoT platform [19] is an example of a platform that provides semantic interoperability across diverse IoT streams in the cloud. The open-source OpenIoT platform[2] enables:

- Sensing-as-a-service and dynamic formulation and deployment of IoT services;
- Semantic unification and interoperability across IoT data streams.

In OpenIoT, all streams are annotated based on the W3C Semantic Sensor Networks (ontology) [20] and hence they can be used in integrated IoT applications.

OpenIoT provides the following services:

- Sensors and internet-connected objects (ICO) deployment and registration;
- Dynamic discovery of sensors and ICOs;
- Visual IoT service definition and deployment;
- IoT service visualization (via mashups);
- Resource management and optimization.

As depicted in Figure 4.4, the OpenIoT architecture is based on three layers (like in the case of Sentillo platform), including [3]:

- *The physical plane:* This deals with the acquisition of observations from the physical world, through either physical or virtual sensors. At this level, an enhanced version of the GSN sensor middleware (called X-GSN) is used in order to integrate data streams from multiple sensors. The X-GSN platform undertakes to transform the data streams to Resource Description Format (RDF) compliant to the OpenIoT ontology, as well as to stream these data to the cloud infrastructure.

2. The platform is available at https://github.com/OpenIotOrg/openiot.

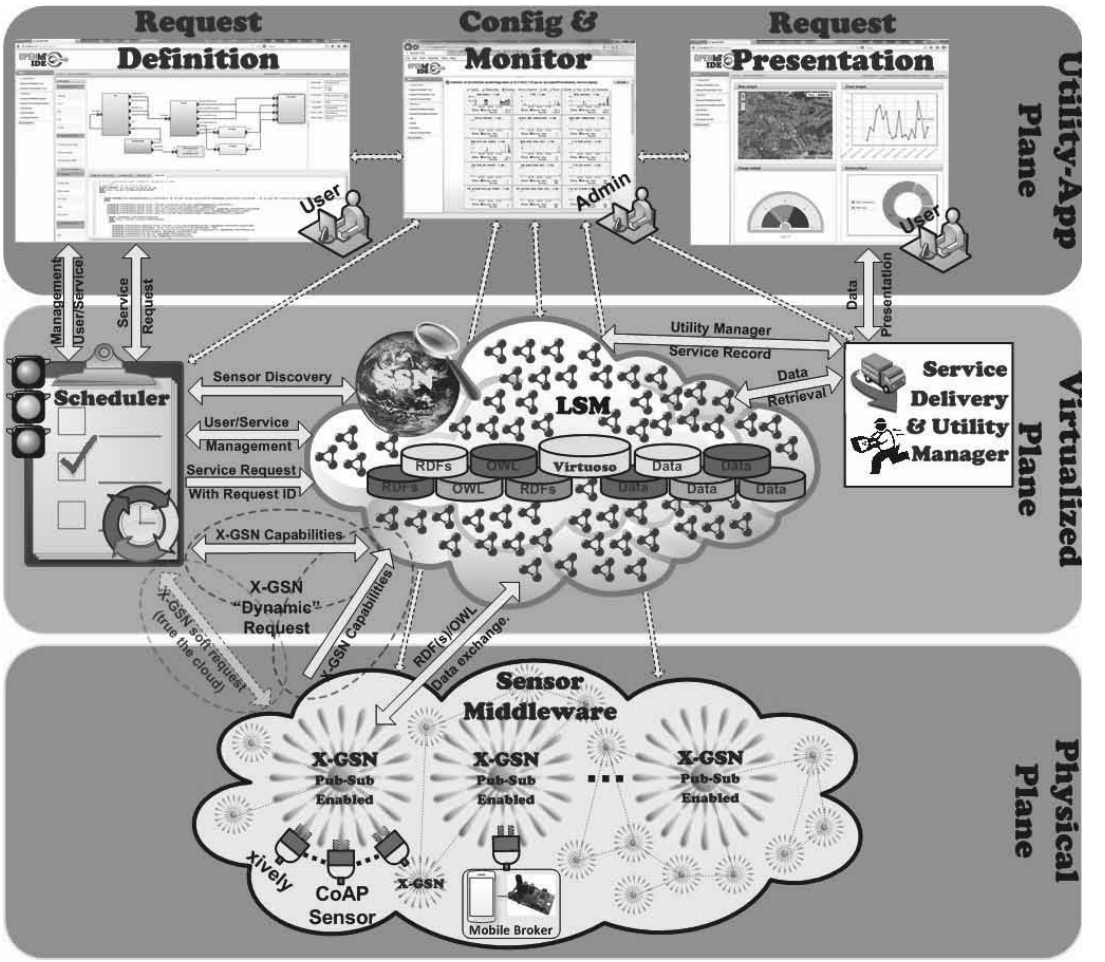

Figure 4.4 Open IoT platform architecture.

- *The virtualized plane:* This provides the means for discovering, accessing, and processing IoT data in a semantically interoperable way. At this layer, data are virtualized, since they are represented in a common way regardless of their location and IoT source. The virtualized plane includes the scheduler component, which deals with the processing of requests for IoT services and the subsequent reservation of resources. It also includes the service delivery and utility manager component, which delivers IoT services according to the discovered sensors and calculates the relevant utilization of resources.
- *The applications plane:* This includes a range of development and configuration tools. The development tools («Request Definition») enable the visual (zero-programming) definition of IoT services, thereby enabling service development without deep knowledge of semantic web technologies (such as SPARQL protocol and RDF query language (SPARQL). At the same time, the configuration tools («Configuration and Monitor») facilitate the monitoring of IoT data and services in the cloud. The applications plane includes

mechanisms and middleware for the visualization of IoT services («Request Presentation»).

OpenIoT has extended the Global Sensor Networks (GSN) middleware [21, 22] with semantic annotation capabilities, in order to ensure that all IoT data streams that connect to OpenIoT, via GSN/X-GSN, following the same semantic format. The extended middleware is called X-GSN. It supports connection of GSN sensors in OpenIoT. It also provides a basis for a framework involving semantics for virtual and physical sensors in GSN, including data representation in RDF and a GSN-Constrained Application Protocol (CoAP) wrapper component which allows accessing sensor data using a RESTful approach and based on the HTTP CoAP, which will be discussed in Chapter 8. This wrapper provides in this way a lightweight RESTful application protocol, which is HTTP-based.

4.4 Summary

The vast majority of IoT applications are nowadays cloud-based. In this chapter, we presented the concept of IoT and cloud integration, including popular cloud-based delivery models for IoT services. We also discussed the limitations of IoT-cloud integration and introduced edge computing as the distributed computing paradigm that can alleviate them. Different types of edge computing architectures and infrastructures have been presented, including fog computing and MEC. Edge computing represents the state of the art in IoT architectures and deployments. It is the distributed computing architecture that is in line with the mandates of standards-based reference architectures for implementing nontrivial IoT systems. This chapter also referred to an indicative, nonexhaustive list of IoT platforms, which are very widely used in the industry.

The integration of IoT data and services in the cloud enables the collection, storage, and processing of large volumes of IoT data. This is a foundation for many IoT analytics applications, including applications that mine IoT data. In Chapter 5, we introduce the characteristics of IoT data, along with tools and techniques for collecting and processing them in the cloud. As such, Chapter 5 is a natural continuation of our IoT and cloud integration discussion.

References

[1] Buyya, R., et al., "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility," *Future Generation Computer Systems*, Vol. 25, No. 6, June 2009, pp. 599–616.

[2] Xen Project, https://xenproject.org/.

[3] Yangui, S., et al., "A Platform as-a-Service for Hybrid Cloud/Fog Environments," *2016 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, Rome, Italy, 2016, pp. 1–7.

[4] Soldatos, J., et al., "OpenIoT: Open Source Internet-of-Things in the Cloud," *Interoperability and Open-Source Solutions for the Internet of Things Cham*, Switzerland:Springer, 2015, pp. 13–25.}

[5] Atlam, H. F., et al., "Integration of Cloud Computing with Internet of Things: Challenges and Open Issues," *2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, Exeter, United Kingdom, June 21–23, 2017, pp. 670–675.

[6] Soldatos, J., et al., "Design Principles for Utility-Driven Services and Cloud-Based Computing Modelling for the Internet of Things," *International Journal of Web and Grid Services*, Vol. 10, No. 2/3, April 2014, pp. 139–167.

[7] Mehedi Hassan, M., B. Song, and E. -N. Huh, "A Framework of Sensor-Cloud Integration Opportunities and Challenges," *Proc. of 3rd International Conference on Ubiquitous Information Management and Communication*, January 15–16, 2009, pp. 618–626.

[8] Liu, F., et al., "A Survey on Edge Computing Systems and Tools," *Proceedings of the IEEE*, Vol. 107, No. 8, August 2019, pp. 1537–1562.

[9] Shi, W., et al., "Edge Computing: Vision and Challenges," *IEEE Internet of Things Journal*, Vol. 3, No. 5, October 2016, pp. 637–646.

[10] El-Sayed, H., et al., "Edge of Things: The Big Picture on the Integration of Edge, IoT and the Cloud in a Distributed Computing Environment," *IEEE Access*, Vol. 6, 2018, pp. 1706–1717.

[11] Satyanarayanan, M., et al., "Edge Analytics in the Internet of Things," *IEEE Pervasive Comput.*, Vol. 14, No. 2, April/June 2015, pp. 24–31.

[12] Bonomi, F., et al., "Fog Computing and Its Role in the Internet of Things," *Proc. of 1st Edition of the MCC Workshop on Mobile Cloud Computing, (MCC '12)*, 2012, pp. 13–16.

[13] Industrial Internet Consortium, "The Industrial Internet of Things Volume G1: Reference Architecture, version 1.8," 2017, http://www.iiconsortium.org/IIRA.htm.

[14] Wang, S., et al., "A Survey on Mobile Edge Networks: Convergence of Computing Caching and Communications," *IEEE Access*, Vol. 5, 2017, pp. 6757–6779.

[15] ETSI, "Multi-access Edge Computing (MEC)," https://www.etsi.org/technologies/multi-access-edge-computing.

[16] Mach, P., and Z. Becvar, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," *IEEE Commun. Surveys Tuts.*, Vol. 19, No. 3, Third Quarter 2017, pp. 1628–1656.

[17] Tran, T. X., et al., "Collaborative Mobile Edge Computing in 5G Networks: New Paradigms Scenarios and Challenges," *IEEE Communications Magazine*, Vol. 55, No. 4, April 2017, pp. 54–61.

[18] Amazon Web Services (AWS), "Amazon DynamoDB," 2020, https://aws.amazon.com/dynamodb/.

[19] OpenIoT Consortium, "OpenIoT," September 2019, openiot.eu.

[20] OCG/W3C, "Semantic Sensor Network Ontology," W3C Recommendation, October 19, 2017, https://www.w3.org/TR/vocab-ssn/.

[21] SourceForge, "Global Sensor Networks (GSN)," March 10, 2014, https://sourceforge.net/projects/gsn/.

[22] Aberer, K., M. Hauswirth, and A. Salehi, "Infrastructure for Data Processing in Large-Scale Interconnected Sensor Networks," *2007 International Conference on Mobile Data Management*, May 1, 2007, pp. 198–205.

[23] Sheng, X., et al., "Sensing as a Service: A Cloud Computing System for Mobile Phone Sensing," *IEEE SENSORS 2012*, Taipei, 2012, pp. 1–4.

CHAPTER 5

IoT Analytics

The majority of IoT applications collect, processe, and analyze data from internet-connected devices. This chapter is devoted to IoT analytics, with an emphasis on the integration of IoT with big data and big data analytics technologies. IoT data are essentially big data due to their volume, velocity, variety, and veracity. This chapter presents these characteristics of IoT data, including their peculiar features and differences from other types of big data (e.g., batch datasets). It also illustrates some big data tools that can be applied and used in the development and deployment of IoT applications.

This chapter starts with a presentation of IoT data as a specific type of big data. It then introduces IoT analytics and presents tools for IoT data streaming. A significant part of the chapter is devoted to the presentation of data streaming engines, which are among the popular data processing tools for IoT analytics applications.

5.1 Analyzing IoT Data

5.1.1 IoT Analytics: The Business Drivers

A great deal of the business value of the IoT paradigm will be created based on the efficient processing of large volumes of IoT data. According to a 2015 study by McKinsey & Company [1], IoT's value potential will be realized based on developments in the following two areas:

- *Interoperability between diverse IoT systems:* This is critically important to capturing maximum value. On average, interoperability is required for 40% of potential value across IoT applications and by nearly 60% in some settings.
- *Effective processing of large volumes of IoT data:* Most IoT data are not currently used. For example, only 1% of data from an oil rig with 30,000 sensors is currently examined. Data used today are mostly used for anomaly detection and control, rather than for optimization and prediction, which will provide the greatest value in the years to come.

As a result, the development and deployment of systems that analyze IoT data in scalable and effective ways are a critical element of the return on investment (ROI) of an IoT system.

5.1.2 Properties of IoT Data

Handling IoT data is a difficult and challenging task, given the characteristics of IoT data streams that differentiate them from other types of data sources. Specifically, nontrivial IoT applications comprise data with one or more of the following properties:

- *Multimodal and heterogeneous:* They stem from a wide array of data sources such as different sensors and other types of internet-connected devices such as wearables and smart CPS.
- *Noisy and incomplete:* In several cases, IoT devices sense and measure physical world properties (e.g., temperature, vibration, images) that are associated with uncertainty and carry noise.
- *Biased:* They usually sample physical phenomena rather than represent them faithfully and provide an exhaustive tracking of them. For instance, acoustic measurements collected from a machinery on a given day may not be representative of the machine's behavior, if collected during a period where the machine was malfunctioning.
- *Time and location-dependent:* IoT data are captured in the scope of a specific time window, while being associated with a certain location (e.g., typically the location of the IoT devices).
- *Dynamic:* IoT devices provide data continually that change at very fine timescales. Several IoT applications process IoT data that change every millisecond (e.g., such as the vibration of a machine) rather than static data that reside in some database (i.e., data at rest).
- *Crowd-sourced:* In cases of crowd-sourcing applications, IoT devices are used by multiple users that contribute information for solving a common problem. For instance, in several cases, multiple users contribute to measuring environmental parameters (e.g., air quality) for smart city applications.
- *Real-time:* Several IoT applications need to analyze results in very short times as a means of fulfilling very strict timing constraints. For example, this is the case of IoT-based security and civil protection applications that provide real-time insights about security incidents.
- *Privacy and security-sensitive:* In several application domains, personal data are processed. As a prominent example, IoT-based healthcare applications use IoT devices (e.g., wearables or internet-connected medical devices) in order to collect information about the healthcare context of the end user. Healthcare applications must collect, store, and process personal data in line with applicable laws and regulations such as the European Union's (EU) General Data Protection Regulation (GDPR).

5.1.3 The IoT Data Life Cycle

IoT analytics applications comprise entire data management workflows that collect, process, visualize, and reuse IoT data. The challenge of IoT analytics lies in the

implementation of such workflows end to end, rather than processing individual IoT data streams. Data collection can be a challenging task, as it requires the implementation of middleware systems that interface to the low-level capabilities of internet-connected objects.

A typical IoT analytic workflow comprises the following phases:

- *Phase 1—Data Collection:* This phase deals with the collection of IoT data from various sources such as sensors, wearables, CPS, and smart objects. The collection process must interface to the various sources and ensure that data streams arrive at the IoT system with appropriate contextual information, including the identification of the stream, as well as time and location information. As part of the data collection phase, the format of the IoT streams is validated. Furthermore, the integrity and the consistency of the data streams are also verified to ensure that the IoT data will be processed appropriately in subsequent phases.
- *Phase 2—Data Analytics:* This phase is devoted to the structuring, storing, and indexing IoT datasets. It also performs correlation of different datasets, compliance checking of the data, and implementation of data governance. Overall, this phase processes the collected IoT data in ways that produce knowledge.
- *Phase 3—Data (Re)use:* This stage deals with the deployment of data in applications, including the provision of appropriate interfaces for accessing them, sharing them across applications, and visualizing them. In this phase, licensing issues and issues associated with secondary (re)use of the data are resolved.

These three phases can be combined in baseline pipelines for data analytics. They are also interacting closely in the scope of various IoT applications. For example, the data use phase (Phase 3) may trigger access to the field for collection of more data (Phase 1). The additional data can be then analyzed further (Phase 2) to extract enhanced insights from the IoT data.

5.1.4 The Vs of Big Data and IoT Data

IoT data processing activities are very similar to the wave of big data technologies. Indeed, IoT data are characterized by the famous Vs that are commonly associated with big data technologies. Specifically, big data systems refer to data processing and management systems, which handle data with one or more of the following characteristics (Vs):

- *Volume:* Very high data volumes, beyond those that can be handled by conventional state-of-the-art data management systems, such as Relational Database Management Systems (RDBMS).
- *Velocity:* Data streams with very high ingestion rates, which cannot be handled by state-of-the-art systems and databases.

- *Variety:* Data featuring extreme heterogeneity in terms of velocities, formats, and semantics. Big data systems and applications are, in most cases, processing data from many different sources such as IoT devices, databases, and social media.
- *Veracity:* Data characterized by uncertainty and unreliability, due to biases, noise, and abnormalities. The latter properties make it very hard to ensure that the data stored and mined is meaningful to the analytics problem at hand.

IoT data feature the four Vs and are different from conventional transactional data. Specifically, large-scale, nontrivial IoT applications deal with:

- *Very high-data volumes:* They collect and process data from numerous IoT devices, which are usually producing new data very quickly. As an example, industrial IoT applications for smart manufacturing collect and process information from many hundreds of automation devices inside a plant. Likewise, connected driving applications handle data from the tens or hundreds of sensors and connected devices that are attached to modern connected vehicles.
- *Many high-velocity streams:* They usually process data from many different streaming sources, such as sensors and CPS. For instance, temperature and vibration sensors continually provide readings. As another example, a mobile robot can provide very frequent readings of its changing location coordinates.
- *A great variety of heterogeneous data sources:* They interface and access data from many different heterogeneous sensors and internet-connected devices. Furthermore, many IoT applications collect and process data from many non-IoT sources such as social networks, files, databases, and data warehouses.
- *High veracity:* Sensor data are noisy and prone to errors. Moreover, most IoT devices are associated with a degree of unreliability and inaccuracy in their operation.

As a result, big data techniques and tools are applicable to IoT analytics. For instance, many IoT analytics applications integrate data mining and machine learning techniques, along with big data databases. In several cases, big data tools are customized to the peculiar characteristics and properties of IoT data.

5.1.5 Properties of IoT Analytics

IoT analytics are characterized by the following special properties that differentiate them other types of data analytics:

- IoT data collection comes at a cost of bandwidth, network, energy, and other resources. It also depends on multiple layers of the network. Therefore, IoT

analytics applications employ resource-aware data analytics methods and cross-layer optimizations.

- IoT data sources go beyond sensor observations. They comprise physical, cyber, and social data. Therefore, IoT analytics solutions work across different CPS, IoT platforms, social media platforms and more.
- Application extracted insights and information should be converted to feedback or actionable information that enables IoT applications to close the loop to the field. For example, in a predictive maintenance application, manufacturing data are processed in order to predict the remaining useful life of a machine. Upon the extraction of this prediction, actuation commands that reconfigure machines (e.g., reducing their rate of operation) or business information systems (e.g., placing service orders in an Enterprise Resource Planning (ERP) system) are given.

Nevertheless, the main distinguishing characteristic of IoT data when compared to other data types is its streaming nature. The latter requires special handling based on different databases and queries, when compared to conventional transactional data.

5.2 Understanding Streaming Data

5.2.1 Streaming Data Nature and Challenges

IoT data are typically streaming data, which are stored and managed in streaming databases. Conventional databases store data in finite, persistent data sets. Nevertheless, they cannot effectively support streaming applications, which need to handle data as multiple, continuous, rapid, time-varying data streams. Overall, conventional (e.g., relational) databases handle finite datasets, while IoT streaming applications handle continuous data. Typical examples of streaming data and respective applications include:

- Social media feeds (e.g., Twitter or LinkedIn posts);
- Auctions and payment transactions;
- Real-time security applications;
- Telecom call records;
- Network monitoring and traffic engineering applications;
- Financial applications for trading that access and process stocks' data;
- Web logs, web analytics, and click-streams;
- Sensor networks data;
- Shop floor data collected in the scope of smart manufacturing processes.

Streaming data are handled by stream systems. The main characteristics of the latter are:

- Their ability to handle multiple streams;
- Their ability to provide support for continuous streams;
- The handling of time-varying streams.

Streaming systems solve the main challenges of working with streaming data including:

- *Continuous queries:* In streaming systems queries are continuous rather than one-time. Hence, they need to be evaluated continuously as stream data arrive, while the answers to these queries should be updated over time.
- *Complex queries:* Queries may be also complex requiring more than "element-at-a-time" processing, as well as more than "stream-at-a-time" processing.

The left side of Figure 5.1 illustrates data storage, persistence and handling in the scope of a traditional database management system, such as an RDBMS. In an RDBMS, data are accessed based on Structured Query Language (SQL) queries. Data can be loaded in the database as part of an Extract-Transform-Load (ETL) process. The right side of Figure 5.1 illustrates the main operation of a streaming database. Instead of loading data based on ETL processes, streaming databases are populated with streaming data that arrive continually in the database. Likewise, queries are answered through a stream query processor, which continually provides data in response to a query. To this end, queries are first registered to the query processor, which undertakes providing data continuously and subsequently to update the result.

Overall, conventional Database Management Systems (DBMS) (e.g., RDBMS) perform batch processing through:

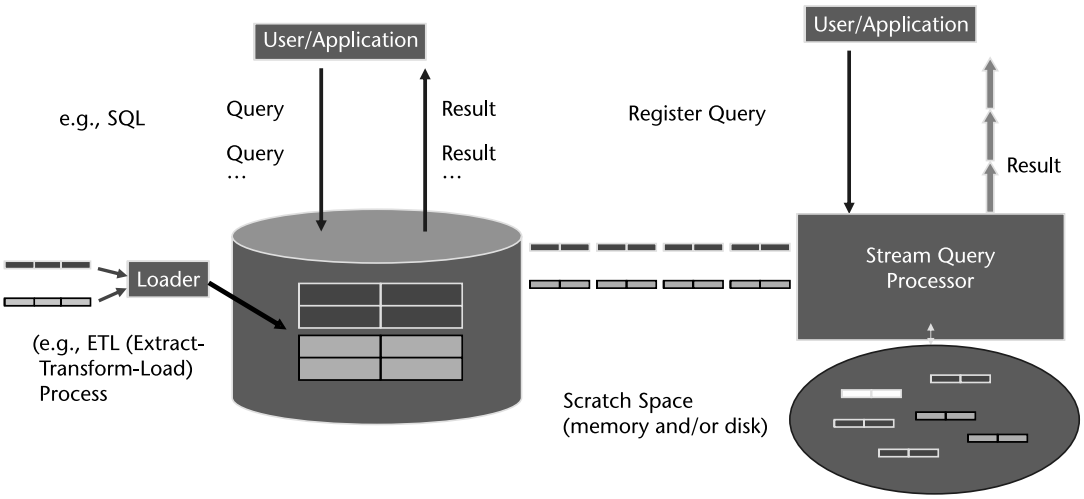

Figure 5.1 Traditional database (left) and streaming database (right).

- Gathering data and processing them as a group at one time;
- Running data processing jobs to completion;
- Handling data that might be out of date.

On the contrary, Data Streaming Management Systems (DSMS) perform real-time processing as follows:

- Processing of data that takes place as the information is entered rather than in a batch model;
- They run forever and in a continuous fashion.

DBMS and DSMS are also different in the following ways:

- DBMS manage persistent relations, while DSMS manage both transient streams and persistent relations.
- DBMS perform one-time queries, while DSMS perform continuous queries.
- DBMS provide random access to data, while DSMS access streams in a sequential way.
- In a DBMS, a query's access plan is determined by the query processor and the physical design of the database, while in DSMS, a plan can be hardly provided due to the unpredictable data arrival and characteristics of the data streams.
- DBMS feature "Unbounded" disk store, while DSMS are bounded by the main memory that is used to handle the streams in a fast way.

IoT analytics applications are mainly about stream processing and hence understanding DSMS and their differences from DBMS is essential to understanding IoT analytics applications.

5.2.2 Streaming Queries

Steaming databases and querying engines process data streams. The main characteristics of a data stream include:

- The type of data, which is reflected in its schema;
- The data distribution;
- The flow rate of the data;
- The stability of its distribution and flow;
- The ordering of the data and other constraints;
- Its synchronization with other streams;
- The distribution of the streams in the case of distributed streaming systems.

Streaming queries are executed over data streams. Their main properties and parameters are as follows:

- *Their answer availability:* Data stream queries can be answered one time or multiple times. They can also be continuous ("standing"), stored or streamed.
- *Their registration time:* Data stream queries can be either predefined or registered in the query engine of the data streaming database in an ad hoc fashion.
- *Their stream access:* Data stream queries can have either arbitrary access to the streams or access restricted to a given time window.

As already outlined, contrary to the evaluation of conventional transactional queries, the evaluation of streaming queries is approximated. Hence, they return an approximate rather than returning an accurate and definite result. The reason for this is that streams come too fast, and hence, finding the exact answer requires unbounded storage or significant computational resources, which are not always available. Also, in the case of ad hoc queries, the history of the data streams is referenced.

The task of approximating the answer to a streaming query is associated with several challenges, including challenges related to the proper selection of sliding windows, sampling rates, and synopses mechanisms. Also, there are different ways for controlling the approximation, which are related to the end users' perception about the query. The approximation process involves various trade-offs, which attempt to balance accuracy, efficiency, and storage requirements.

A streaming engine should be able to handle multiple queries, while adapting the handling of each one to changing conditions. Adaptivity takes place as a result of the long-running nature of the queries, but also as a result of their fluctuations in terms of arrival times and data characteristics. Furthermore, query loads are evolving as streams arrive at different rates. The goal of the adaptation process is to adapt the allocation of resources used to answer a query (i.e., memory and computation), but also to adapt the query execution plan, in response to changes in the arrival characteristics of the data streams and the system load changes.

Streaming systems must also handle a large number of continuous queries, which are long-running and use shared resources. This makes the problem of resource optimization even more challenging, as optimal resource allocation should not only take place at the level of one query, but rather for multiple queries.

5.2.3 Distributed Streaming Systems

The main characteristic of distributed streaming systems is that they can optimize resource allocations across multiple concurrent streaming queries. Specifically, a distributed streaming system is characterized by:

- Its ability to consider and handle many physical streams as one logical stream;
- Its ability to monitor, manage, and correlate streams that physically reside at distributed servers;
- Control of many streams by a fewer number of servers (e.g., one server may be assigned to control a sensor network).

Likewise, one of the main challenges faced by distributed streaming systems implementations is the need to move processing to streams, rather than moving streams to a given processor. Moreover, there is a need to resolve bandwidth related trade-offs, in order to optimize the overall performance of the system. This optimization takes into account the available computing and network resources.

A distributed streaming system is fed with multiple input data streams. The core engine of the system (i.e., its query engine) applies various operators over the distributed streams, in order to answer streaming queries in the most efficient way. Operators are applied on physical streams in line with a computed query plan. The output of the query is typically an output stream, which is provided to the applications that are supported by the distributed streaming system. The operators of the DSMS are offered with interfaces for defining and executing queries, while at the same time monitoring query execution and fine-tuning run-time parameters. Overall, the main elements of a DSMS include:

- *Query planning and execution infrastructure:* This provides support for executing operators, synopses of the streams, and queuing of the data that comprise the streams.
- *Memory management infrastructure:* This provides dynamic allocation of memory to buffers, queues, and synopses functions. It also resolves accuracy versus memory use trade-offs. Moreover, it caters for adapting operators to memory reallocation operations.
- *Scheduling functionality:* This enables the handling of variable-rate input streams, as well as varying operator and query requirements.

5.2.4 Streaming Systems and IoT Data Processing Architectures

Data streaming is an essential part of IoT data processing architectures. Traditional databases and ETL pipelines are largely based on batch data operations. As such, they are not adequate and efficient in coping with volume, speed, and variety of data that are produced by IoT networks and devices. IoT data processing architectures must be capable of handling streaming data on a massive scale, while at the same time allowing users to react to data exactly when the data are generated. Such architectures reduce operational complexity and help to overcome connectivity issues.

Handling streaming data is usually performed at the edge of the network, in line with the edge computing paradigm. IoT data are usually arriving at edge nodes with high ingestion rates and must be processed instantly at the edge. Edge processing is fundamental for economizing storage and bandwidth, but also for taking decisions in real time. Edge processing can become challenging in cases of poor connectivity, as data might arrive out of order. Out-of-order data make the processing and analysis of streaming data more challenging. There are many platforms that support handling big data with an emphasis on IoT data and streaming. These platforms facilitate the handling of IoT data and IoT analytics at scale and, as such, are integral elements of most large-scale IoT infrastructures.

5.2.5 Evolution of Stream Processing Platforms and Technologies for IoT

The importance of streaming analytics for IoT and big data applications has given rise to several research and industry efforts that produced high-performance stream processing engines. A first generation of streaming analytics applications used centralized stream processing engines such as Aurora [2] and TelegraphCQ [3]. These engines provided window-based query operators that execute continuous queries over relational data streams. They supported the relational query model (e.g., the Continuous Query Language (CQL) [4]) but did not support parallel data processing. The increase of stream rates and query complexity led to the development of another generation of stream processing engines, which were distributed and could harness the processing power of a cluster of stream processors. Typical examples of such systems are Borealis [5] and InfoSphere Streams [6], which permit interoperator parallelism for continuous queries (i.e., one query can be executed on multiple machines). Such systems exploit task-parallelism (i.e., they execute different operators on different machines and allow the execution of many different continuous queries in parallel).

Systems such as InfoSphere Streams support intra-query parallelism, which specifies stream connections, but management and configuration are manual. This poses limitations for big data applications, where a single continuous query must, in several cases, process a large volume of streaming data. To alleviate this limitation, stream processing engines that focus on intra-query parallelism emerged. The latter parallelize the execution of individual query operations. Typical examples include StreamCloud [7] and the popular Apache S4 and Apache Storm systems, which express queries as directed acyclic graphs with parallel operators interconnect by data streams. However, these systems cannot scale out the computation at run time and therefore are not effective in supporting unknown big data analytics jobs when the computational resources needed are not known ahead of time. To this end, systems such as Spark Streaming [8] that parallelize streaming queries by running them on the Spark distributed dataflow framework using micro-batching have emerged. Micro-batching permits execution of streaming computations as a series of short-running Spark job that output incremental results based on the most recent input data. Nevertheless, these execution models have limitations when it comes to supporting sliding windows in the streaming process.

In recent years, the Apache Flink engine [9] incorporated a distributed dataflow framework that can execute data-parallel batch and streaming processing jobs on the same platform. Computation is described as dataflow graphs, which are optimized using existing database techniques. This is the reason why Apache Flink and Apache Spark are currently two of the most popular streaming engines for big data systems. However, these platforms assume that stream processing operators are stateless. While this simplifies scalability and failure recovery, it is a setback to expressing complex analytic tasks such as data mining and machine learning algorithms that need to refine a model incrementally. To address this problem, some of the state-of-the-art stream processing engines adopt a stateful stream processing model. For example, this is the case with Apache Samza [10] and Naiad [11], which execute streaming operators in a data-parallel fashion. More recent streaming engines implement the concept of Stateful Dataflow Graphs (SDGs), which

are graphs containing vertices that are data-parallel stream processing operators with arbitrary amounts of mutable in-memory state, and edges that represent the stream. SDGs can be executed in a pipelined fashion and have a low processing latency. When processing SDGs, the machines in a cluster take checkpoints of the in-memory state of processing operators, which are stored to disk. Therefore, in case a machine in the cluster fails, the failed operator instances can be restored on other machines, recovering their state from the most recent checkpoint and reprocessing buffered stream data in order to restore the operator's state in a way that ensures that it is up-to-date.

Recently, Industrial IoT (IIoT) vendors and solution integrators also make extensive use of the Kafka Streams framework [12], which incorporates many of the previous listed functionalities and addresses many of the stream processing challenges in IIoT environments. It supports event-at-a-time processing with millisecond latency, which alleviates the limitations of micro-batching. Moreover, it provides the means for stateful processing including distributed joins and aggregations. Furthermore, it supports distributed processing and failover, while at the same time offering a convenient Domain Specific Language (DSL) for defining queries.

It is quite common for IIoT environments to deploy different streaming engines and toolkits (e.g., Spark, Kafka, and Storm). Hence, a need for routing data streams from different streaming middleware platforms to consumers and applications is arising. There is a need for a meta-streaming engine for streaming data routing and processing, which can make sure that data streams acquired by different streaming engines are delivered in the target application. The concept of such a meta-level streaming engine is perfectly in line with edge computing systems, as the latter are likely to combine different streaming middleware platforms in their edge nodes. Meta-level streaming functionalities can be defined by means of a DSL [13, 14], which provides versatility in configuring domain-specific data operations in ways that ease programming and lower development times [15].

The next section provides more information about some of the most used platforms for IoT streaming, such as Kafka, Spark, Storm, and Flink.

5.3 Popular Big Data Management and Distributed Streaming Platforms

5.3.1 Big Data Storage and Management: The Hadoop Distributed File System (HDFS)

Big data exceed the data storage and data management capacity of conventional relational databases, even when the latter are deployed as part of distributed configurations such as computing clusters. This has motivated the development of data storage and management systems that are tailored to handling big data.

Big data storage and management systems are based on large-scale distributed systems that facilitate resource sharing across multiple physical systems. The most fundamental infrastructure for persisting and managing big data is a distributed file system (i.e., a system that can handle very large volumes of data in a scalable and resilient way). The concept of a distributed file system is inherited from the concept

of a file system. A file system is typically responsible for the organization. storage, retrieval, naming, sharing, and protection of files. It is also responsible for controlling access to the data, while providing support for low-level operations such as the buffering of frequently used data. In line with these properties, the goal of a distributed file system is to allow users of physically distributed computers to share data and storage resources based on a common file system.

The most prominent example of a distributed file system for big data is the Hadoop Distributed File System (HDFS) [16], which is part of the Apache's open-source Hadoop framework. Hadoop provides the means for the distributed processing of large data sets across clusters of computers using simple programming models. Hadoop's main characteristic is that it scales up from single servers to thousands of machines, each offering local computation and storage. Apart from HDFS, Hadoop comprises the following components:

- *Hadoop YARN:* This is Hadoop's framework for job scheduling. It supports cluster resource management [17].
- *Hadoop MapReduce:* This is a YARN-based system for parallel processing of large data sets [18]. While MapReduce was one of the first parallel processing systems, nowadays other execution engines are increasingly in use such as Apache Spark.

HDFS is the most used Hadoop component. It is fault-tolerant and reliable and provides high-throughput access to application data. The main principle behind the operation of Hadoop is that it moves computation to the place of data, as a means of enabling processing of huge amounts of data (e.g., petabytes) in a scalable way.

HDFS distributes the data and processing across clusters of commonly available computers (i.e., commodity hardware). Commodity hardware is preferred over expensive, ultra-reliable hardware, since it offers a better throughput/price (or performance/price) ratio, which is more important than peak performance. Moreover, the handling of very large amounts of data requires many distributed computers, which means that even with reliable hardware, computer failures will be statistically unavoidable. In this context, HDFS ensures data reliability at the software level as follows:

- Large files (i.e., typically gigabytes/terabytes) are broken into fixed-size blocks (i.e., typically 64 MB or 128 MB) and distributed across different machines.
- Files are also replicated across multiple machines in order to enable handling of hardware failures. Both the block size and the replication factor are configurable per file. HDFS maintains automatically multiple copies of data, while providing the means to redeploying computing tasks whenever failures occur.
- Checksums for corruption detection and recovery are supported.
- Data operations can continue as nodes or racks of nodes are added or removed.

HDFS is also optimized for fast batch processing. The location of the data is exposed to allow computations to be moved to data. Specifically, specialized nodes of the Hadoop infrastructure (called NameNodes) keep track of where the various blocks of a given file are stored. Likewise, the various blocks of the files reside in other nodes/computers of the infrastructure (called DataNodes). A typical operation on a big data file involves querying the NameNode about the locations (i.e., the DataNodes) of the blocks of the file and subsequently of direct access to the appropriate DataNodes (Figure 5.2). Note that HDFS knows which servers are closest to the data to reduce network traffic. To this end, it comprises a Job tracker, which schedules jobs to task trackers with an awareness of the data location. In this way, HDFS reduces amount of network traffic and prevents unnecessary data transfer. Hence, location awareness is key to reducing job-completion times when running data-intensive jobs. Overall, by moving computations to the data and by parallelizing data processing operations, HDFS can achieve very high aggregate bandwidth and overall very good efficiency.

NameNodes and DataNodes are two of the main components of HDFS. Specifically:

- A NameNode manages the file system namespace and controls read and write access to files. It also manages block replication and keeps metadata about the files in its main memory to allow for fast access to them. The type of metadata that are kept in the NameNode includes the list of the files, the namespaces for the files, and their blocks/chunk namespaces, the list of blocks of each files, the location of the blocks and their replications (replicas), and the attributes of the various files.
- DataNodes serve read and write requests from clients. They also perform replication tasks for the blocks of the various files based on coordination and instructions provided by a NameNode. They store data in their local file system, along with relevant metadata (e.g., a checksum). Moreover, they report periodically about the status of the file blocks to the NameNode. Likewise, they also send heartbeats to the NameNode to facilitate the detection of node failures.

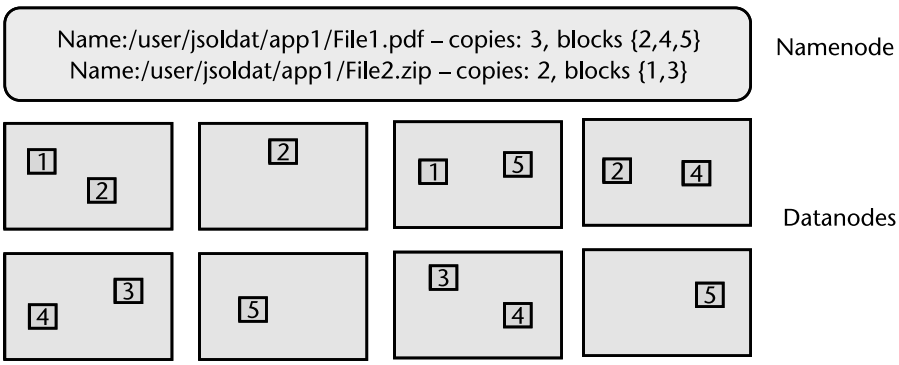

Figure 5.2 HDFS NameNodes and DataNodes concepts.

By default, HDFS stores data on three DataNodes: two on the same rack, and one on a different rack. The NameNode tracks names and locations of the blocks (i.e., it acts as a naming and directory service for blocks).

HDFS is the one of the popular infrastructures for big data storage and management. One or more Hadoop clusters can be combined in a single data management platform, which is called data lake and is used to process and store nonrelational data. In practice, data lakes store and process big data from heterogeneous sources like clickstream records, JavaScript Object Notation (JSON) objects, images, social media posts, sensor data, and IoT data.

Over the years, various Hadoop alternatives have emerged, including, for example, Apache Spark, Apache Storm, and Google's BigQuery. Some of them are very well suited for IoT, since they provide support for the special characteristics of IoT data that differentiate it from other types of big data, notably the streaming nature of IoT data.

5.3.2 Apache Kafka

Apache Kafka is one of the key pillars of robust IoT data processing at scale. Kafka was initially developed at the popular LinkedIn professional networking platform and later became part of the Apache project. It is an open-source software platform that is designed to handle massive amounts of data streams with high ingestion rates [19]. Specifically, it is a distributed stream processing platform for big data systems. Its main functionalities resemble enterprise message systems: Kafka stores streams of records and eases the configuration and deployment of pipelines across different source-destination pairs. The Kafka platform provides fault tolerance: every message written in Kafka is persisted and replicated to peer brokers for fault tolerance. As such, it is also appropriate for supporting messaging for business-critical operations.

Overall, Kafka enables the following three types of functionalities and their combination:

- Reading and writing streams of data such as a messaging system (i.e., in an asynchronous fashion);
- Processing data streams in real time, through writing scalable stream processing applications that respond to events;
- Storing streams of data safely in a distributed, replicated, and fault-tolerant cluster.

Based on these functionalities, Kafka is commonly used for real-time analytics, as it can concurrently handle multiple live data feeds. In practice, for large-scale big data deployments, Kafka is commonly used as a gateway to data processing pipelines that are later deployed in the scope of data centers within clusters powered by other platforms like Apache Storm, Apache Spark, and Apache Hadoop. Likewise, in the more specific context of IoT applications, Kafka provides the means for robust data processing prior to feeding the data to a cluster for further analysis (e.g., machine learning over IoT data). The concept is illustrated in Figure 5.3. Based

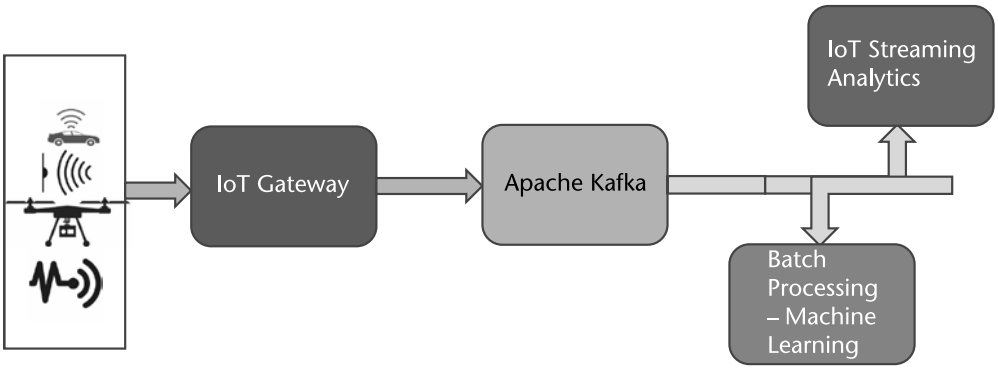

Figure 5.3 Typical use of Kafka distributed streaming in IoT applications.

on such configurations, Kafka collects and stores data streams in a reliable, fault-tolerant fashion, allowing other platforms and frameworks to focus on other tasks such as storing and processing very large amounts of data.

5.3.3 Apache Spark

As already outlined, Hadoop is not designed to handle streaming IoT data. However, its infrastructure and associated ecosystem of tools provide a basis for the operation of other streaming engines, such as Apache Spark. Apache Spark is an open-source software that enables unified big data analytics over different types of data [20], including IoT streaming data. It is very fast and achieves high performance for both batch and streaming data. Moreover, it provides a library for different types of analytics tasks, including SQL analytics, streaming analytics, and data mining based on machine learning. Also, it offers versatility for developers since it supports many popular programming languages for big data such as Java, Python, R, and Scala.

Figure 5.3 has already illustrated the possible role of Apache Spark in the IoT analytics pipeline. In most IoT use cases, real-time streams are collected from edge devices, IoT gateways, and other sources. Following a preprocessing through a Kafka deployment, these data can be processed by Spark Streaming applications. Apache Spark can then generate derived streams for further processing, including data aggregation or trigger even triggering of other real-time events. Spark has the virtue of supporting both stream and batch processing. The latter is also possible in IoT applications, as massive amounts of IoT data can be collected in order to be analyzed at coarse time scales rather than in real time.

Apache Spark's key construct is the Resilient Distributed Dataset (RDD). RDDs represent data or transformations on data and provide abstractions for processing them. RDDs can be created from Hadoop Input Formats (e.g., HDFS files) or through transforming other RDDs. Actions can be applied to RDDs in order to force calculations and return values. RDDs operate based on a lazy evaluation mode, which means that nothing can be computed until an action requires it. RDDs are best suited for applications that apply the same operation to all elements

of a dataset. However, they are less suited for applications that make asynchronous fine-grained updates to shared state. RRD can be persisted (cached) and later unpersisted. Each node stores any partitions of it that it computes in memory. It also reuses them in other actions on that dataset or datasets derived from it. Hence, future actions are much faster (e.g., over 10 times faster).

5.3.4 Apache Storm

Apache Storm is another popular streaming middleware platform. It is a free and open-source distributed real-time computation system, which facilitates reliable processing of unbounded streams of data. Storm deals with real-time processing much in the same way that Apache Hadoop deals with batch processing. Storm enables fast, scalable, reliable, and fault-tolerant stream processing [21].

Storm leverages components of the wider Hadoop ecosystem, including Nimbus that allows Storm to take advantage of IaaS cloud services and Zookeeper that provides configuration, naming, and distributed synchronization services. The Nimbus Platform enables Storm users to leverage Nimbus, OpenStack, Amazon, and other clouds.

Storm is positioning in IoT analytics pipelines in ways similar to Spark. Nevertheless, Storm is very efficient for streaming applications only and does not provide the versatility required to deal with other types of big data workloads and applications (e.g., batch big data analytics).

5.3.5 Apache Flink

Apache Flink is another open-source stream processing framework [22]. It provides low latency and high throughput in data operations. Flink's operations include handling of stateful and distributed transactions. It also provides low-latency streaming ETL operations, which have much higher performance than traditional ETL for batch datasets. Flink is event-time aware: this means that events stemming from the same real-world activity could arrive out of order in the system, but Flink can maintain the correct order in such cases. As earlier outlined, this can be very important for IoT use cases that involve poor connectivity or devices that lost their connection to the IoT platforms, as this can cause data to arrive out of order.

Flink deployments comprise an infrastructure layer (i.e., the computing infrastructure that supports Flink). Flink can run over a single Java Virtual Machine (JVM) or a YARN cluster or even a cloud infrastructure. Over these infrastructures, the Flink run time provides the distributed streaming dataflow capabilities, while offering two APIs on top of it: one for data streaming operations (i.e., DataStream API), and another for batch processing (i.e., DataSet API). Over these APIs, Flink offer toolkits and libraries for Complex Event Processing (CEP), relational database management, machine learning (FlinkML), and Graph Processing (Gelly).

5.4 Summary

This chapter presented the main characteristics of IoT data and of IoT analytics applications. It introduced IoT analytics applications as a specific type of big data applications that feature the Vs of big data. However, it also highlighted the streaming nature of IoT, along with the differences between stream processing and conventional batch transaction processing.

Nowadays, IoT developers and solution integrators are offered with a host of platforms and tools for processing IoT streams in scalable and high-performance ways. This chapter presented some of the most used platforms for processing IoT data streams, including Kafka, Spark, Storm, and Flink. These platforms facilitate the implementation of machine learning algorithms for mining IoT data, which is the subject of Chapter 6.

References

[1] McKinsey Digital, *Unlocking the Potential of the Internet of Things,* June 1, 2015, https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/the-internet-of-things-the-value-of-digitizing-the-physical-world.

[2] Abadi, D., et al., "Aurora: A New Model and Architecture for Data Stream Management," *The VLDB Journal*, Vol. 12, No. 2, August 2003, pp. 120–139.

[3] Chandrasekaran, S., et al., "TelegraphCQ: Continuous Dataflow Processing," *Proc. of 2003 ACM SIGMOD International Conference on Management of Data (SIGMOD '03)*, ACM, New York, NY, 2003, pp. 668–668.

[4] Arasu, A., S. Babu, and J. Widom, "The CQL Continuous Query Language: Semantic Foundations and Query Execution," *The VLDB Journal*, Vol. 15, No. 2, June 2006, pp. 121–142.

[5] Ahmad, Y., et al., "Distributed Operation in the Borealis Stream Processing Engine," *Proc. of ACM SIGMOD International Conference on Management of Data (SIGMOD '05)*, ACM, New York, NY, 2005, pp. 882–884.

[6] Biem, A., et al., "IBM Infosphere Streams for Scalable, Real-Time, Intelligent Transportation Services," *Proc. of ACM SIGMOD International Conference on Management of Data (SIGMOD '10)*, ACM, New York, NY, 2010, pp. 1093–1104.

[7] Gulisano, V., et al., "StreamCloud: An Elastic and Scalable Data Streaming System," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 23, No. 12, December 2012, pp. 2351–2365.

[8] Zaharia, M., et al., "Apache Spark: A Unified Engine for Big Data Processing," *Communications of the ACM*, Vol. 59, No. 11, November 2016, pp. 56–65.

[9] Carbone, P., et al., "Apache Flink™: Stream and Batch Processing in a Single Engine," *IEEE Data Eng. Bull.*, Vol. 38, 2015, pp. 28–38.

[10] Noghabi, S., et al., "Samza: Stateful Scalable Stream Processing at Linked," *Proc. VLDB Endow.*, Vol. 10, No. 12, August 2017, pp. 1634–1645.

[11] Murray, D., et al., "Naiad: A Timely Dataflow System," *Proc. of Twenty-Fourth ACM Symposium on Operating Systems Principles (SOSP '13)*, ACM, New York, NY, 2013, pp. 439–455.

[12] Kreps, J., N. Narkhede, and J. Rao, "Kafka: A Distributed Messaging System for Log Processing," *6th International Workshop on Networking Meets Databases (NetDB)*, June 2011.

[13] Kefalakis, N., A. Roukounaki, and J. Soldatos, "Configurable Distributed Data Management for the Internet of the Things," *Information*, Vol. 10, 2019, p. 360.

[14] Kosar, T., S. Bohrab, and M. Mernika, "Domain-Specific Languages: A Systematic Mapping Study," *Information and Software Technology*, Vol. 71, March 2016, pp. 77–91.

[15] Earley, S., "Analytics, Machine Learning, and the Internet of Things," *IT Professional*, Vol. 17, No. 1, January-February 2015, pp. 10–13.

[16] Shvachko, K., et al., "The Hadoop Distributed File System," *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, Incline Village, NV, 2010, pp. 1–10.

[17] Vavilapalli, V. K., et al., "Apache Hadoop YARN: Yet Another Resource Negotiator," *Proc. of 4th Annual Symposium on Cloud Computing (SOCC '13). Association for Computing Machinery*, New York, NY, Article 5, 2013, pp. 1–16.

[18] Dean, J., and S. Ghemawat. "MapReduce: Simplified Data Processing on Large Clusters," *Commun. ACM*, Vol. 51, No. 1, January 2008, pp. 107–113.

[19] Kreps, J., N. Narkhede, and J. Rao, "Kafka: A Distributed Messaging System for Log Processing," *Proc. of 6th International Workshop on Networking Meets Databases (NetDB)*, Athens, Greece, 2011, pp. 1–7.

[20] Zaharia, M., et al., "Spark: Cluster Computing with Working Sets," *HotCloud 2010*, June 2010.

[21] van der Veen, J. S., et al., "Dynamically Scaling Apache Storm for the Analysis of Streaming Data," *2015 IEEE First International Conference on Big Data Computing Service and Applications*, Redwood City, CA, 2015, pp. 154–161.

[22] Katsifodimos, A., and S. Schelter, "Apache Flink: Stream Analytics at Scale," *2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW)*, Berlin, 2016, pp. 193–193.

CHAPTER 6

Mining IoT Data

Chapter 5 introduced IoT data and IoT analytics, including their main properties. It also presented platforms and techniques for the scalable collection, storage, persistence, and analysis of IoT data. The analysis of IoT data can be performed based on simple rules (e.g., if-then-else types of rules), such as rules that define thresholds, conditions, and constraints on IoT data, along with actions that should be undertaken when these conditions are met. Nevertheless, the full potential of IoT analytics lies in mining IoT data towards extracting knowledge based on it and driving more automated decisions. Data mining is performed based on the statistical analysis of datasets and machine learning techniques, which fall in the realm of artificial intelligence (AI). This chapter focuses on the presentation of techniques for mining IoT data, including machine learning techniques for IoT data. Hence, it is devoted to methods for advanced analytics of IoT data, which is a natural add-on to the collection, storage, and streaming analytics techniques presented in Chapter 5 [1].

This chapter presents the main principles that drive data mining for IoT systems and enable the discovery of knowledge from IoT dataset. The challenging nature of IoT data mining is also discussed. Moreover, this chapter provides an overview of popular machine learning techniques, including supervised and unsupervised machine learning. Furthermore, the classification of machine learning techniques in traditional machine learning, deep learning, and reinforcement learning techniques is provided.

6.1 IoT Data-Mining Activities

6.1.1 Overview

The process of mining and extracting knowledge from datasets is multidisciplinary. It typically requires knowledge and skills in the following areas:

- *Databases:* Data mining requires collecting, consolidating, and managing data within databases. It also asks for knowledge on how to read and write data to different types of database management systems such as Structured Query Language (SQL) databases, no-SQL databases, data warehouses, and data lakes. In most cases, data science and IoT data analytics teams will have to transform the data from one format to another, prior to feeding it to a

machine learning algorithm or prior to persisting the outcomes of knowledge discovery to a database.

- *Machine learning:* Machine learning refers to the study and application of algorithms and statistical models over datasets, towards identifying knowledge patterns in the data. In principle, machine learning algorithms extract rulesets and other types of knowledge based on the processing of historic datasets. They enable machines to learn based on past observations and domain knowledge about how to interpret these observations. Machines learn in ways quite similar to how humans learn from past experiences. There are different types of machine learning techniques, including supervised and unsupervised machine learning, as well as deep learning and reinforcement learning. These different types of machine learning techniques will be explained later. Note also that machine learning is a subset of AI. That is the reason why machine learning is referred to as AI in several contexts.
- *Statistics:* The statistical properties of data-driven processes can also provide the means for extracting knowledge from datasets. Specifically, statistics enable data scientists to express assumptions about the expected behavior and the evolution of processes. These assumptions can then be validated based on real-life datasets. As already outlined, machine learning models are also developed based on statistical models about business processes. Hence, the boundaries between statistics and machine learning in knowledge extraction are often blurred. The same piece of knowledge that can be produced based on assumptions about the statistical properties of a process can be, in most cases, derived based on some machine learning model as well.
- *Visualization:* The proper presentation of the derived knowledge is a key prerequisite for improving the knowledge extraction processes, but also for taking advantage of the knowledge in the scope of a real-life IoT deployment. Therefore, data-mining and data science teams need to possess data visualization skills as well. While some basic visualizations (e.g., bar charts, line charts, pie charts) are very straightforward and can be developed easily, other visualizations can be very challenging. This is the case when visualization of very large datasets (i.e., big data) is needed, as well as in cases where complex data relationships need to be presented to end users.

Overall, data-driven knowledge discovery from IoT datasets is very challenging and requires multidisciplinary teams. The multidisciplinary nature of data mining is evident in the scope of most standardized data-mining processes.

6.1.2 Standardized Data-Mining Processes

The selection of a proper machine learning and data-mining model for discovering knowledge based on IoT data is very challenging. This is because the knowledge extraction process needs to consider the context of the IoT application and to analyze a wealth of IoT data in different settings and environments. To this end, disciplined methodologies for analyzing the data and evaluating the performance of alternative

data-mining models are employed. Some of the most popular methodologies for analyzing and mining datasets are [2]:

- *Cross Industry Standard Process for Data Mining (CRISP-DM) process [3]:* This is an open process that describes the sequence of activities that are most performed by data scientists.
- *Knowledge Discovery in Databases (KDD):* This is another process for discovering knowledge within collections of data, which is widely used in marketing, fraud detection, telecommunication, and manufacturing applications.
- *Sample, Explore, Modify, Model, and Assess (SEMMA) process:* This is a list of sequential steps for knowledge extraction. SEMMA has been developed by SAS Institute, one of the largest producers of statistics and business intelligence software.

These three methodologies and processes are all iterative and cross-sector, which means that they are commonly used in various application domains. Furthermore, all of them provide the means for evaluating the performance of a machine learning model on the supplied datasets, prior to the final selection and field deployment of a data-mining algorithm. These methodologies have not been developed exclusively for IoT data and applications. Rather, they are general-purpose knowledge discovery models, which are applied and used for all types of data-mining problems, including mining IoT data.

6.1.3 A Closer Look at CRISP-DM

CRISP-DM is probably the most popular of the three previously outlined processes. CRISP-DM is an iterative methodology comprising six major phases. The phases are sequential in the sense that each one is based on the outcomes of the previous one. Nevertheless, due to the iterative nature of the methodology, it is possible and, in most cases, required to revert from one phase to a previous one. The six phases of CRISP-DM are illustrated in Figure 6.1 and are described later.

6.1.3.1 Business Understanding

The business understanding phase is the initial phase of the methodology. It sets the scene and decides the scope of the data-mining activities. It establishes the requirements and ultimate goals of the data-mining process, including what is the expected result. To this end, the target business question must be formulated. This could be the prediction of a machine's end of life based on vibration and ultrasonic data or even the prediction of traffic flows within an urban traffic route. Both examples are typical data-mining problems that take advantage of IoT data. In addition to formulating the target business question, a preliminary plan to resolving this question is developed, including the datasets to be used and the models that should be explored.

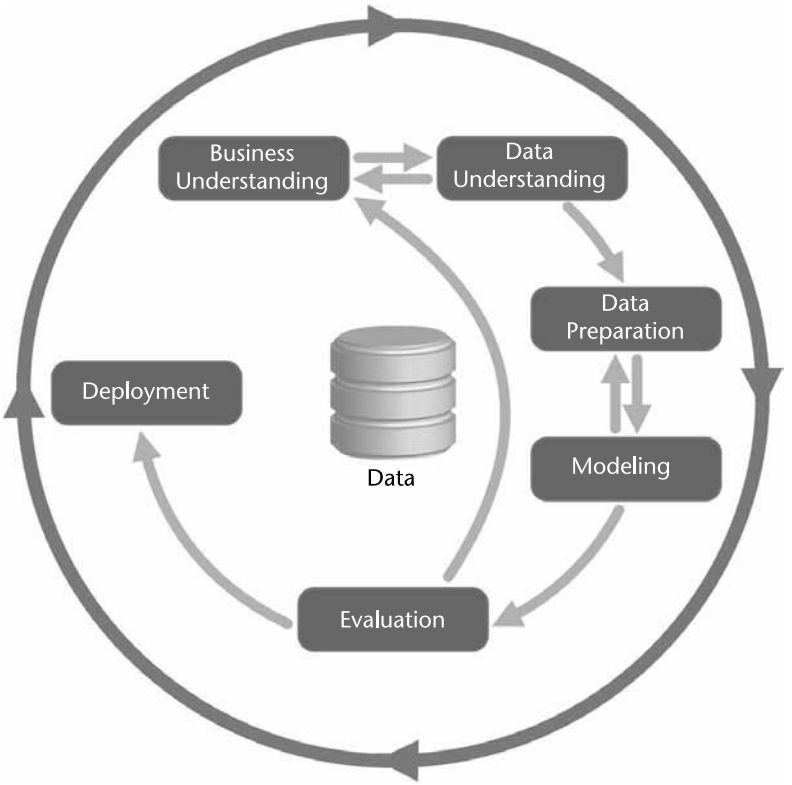

Figure 6.1 The phases of the CRISP-DM methodology [3]. (Image by Kenneth Jensens, https://commons.wikimedia.org/wiki/File:CRISP-DM_Process_Diagram.png.)

6.1.3.2 Data Understanding

As part of the data understanding phase, the collected datasets are reviewed. For the success of the data-mining process, it is very important to inspect the available datasets to identify data quality problems, but also to understand which models could be effective and which not. Even though every problem is different, experienced data scientists can prioritize the methods to be tested and evaluated simply by reviewing the available data. The review may also involve visualization of the raw datasets, as well as calculation of their statistical properties. The latter could help data scientists to identify candidate machine learning models and other techniques that could potentially work for the problem at hand.

As a prominent example, in the scope of an energy demand forecasting application, a data scientist would like to observe the statistical properties of data from smart meters and meteorological stations, in order to evaluate their appropriateness for the prediction task. Likewise, these data could be also used to drive the selection of predictive analytics algorithms for the problem at hand.

6.1.3.3 Data Preparation

In the data preparation phase, the datasets to be used for extracting and evaluating the data-mining model are prepared. This involves several transformations that are

applied over raw data from IoT data sources (e.g., sensors, smart meters, moving objects), as well as over non-IoT data sources (e.g., conventional transactional databases and business information systems). The data preparation involves for example filtering of the datasets (e.g., selecting specific attributes), transforming them in different formats, combining datasets (e.g., joining datasets from different sensing modalities), as well as cleaning them (e.g., getting rid of empty or incomplete fields). For instance, JSON data from a sensing device might have to be converted to the Comma Separated Values (CSV) format to be fed to a data-mining algorithm. Likewise, some columns of the CSV might need to be erased, while other would have to be converted to other types (e.g., from numerical values to strings). The ultimate objective of this phase is to ensure that the data are ready to be loaded and used by data modeling tools.

6.1.3.4 Modeling

There is a variety of machine learning models that can be used for classification, prediction, rules extraction, and knowledge generation. Data scientists select some of these models, typically the ones deemed most suitable for the problem at hand. The purpose of the modeling phase is to apply the selected models on the available data and to calibrate them by tuning their parameters. However, each different model to be applied will most likely need input data in different formats. Therefore, it is very common for data scientists to revisit the data preparation phase to prepare alternative datasets. Later we outline a wide array of machine learning techniques, which are typically used and fine-tuned as part of this modeling step.

6.1.3.5 Evaluation

Following the development of the data model(s) in the previous phase, the evaluation phase performs a thorough evaluation of the operation of the selected models against the target objectives. The evaluation is conducted in terms of different parameters, including the ability of the model to extract the targeted knowledge, its accuracy, its scalability, its "explainability," and its "interpretability" (i.e., whether it is easy to explain its operation). As a prominent example, during the evaluation phase, a model can be tested against its ability to produce traffic productions that are very close to the known traffic on a given transportation route.

Most importantly, the evaluation phase audits a model against its ability to confront the business problem that was formulated in the business understanding phase. The latter may require that the model fulfills several functional and non-functional requirements (e.g., execution speed that is below a certain threshold). This assessment of the model against the set business objectives drives the final decision on whether a model can be deployed in production or not. In cases where none of the tested models meets the target business objectives, the data-mining team has the option to go back from this phase to the business understanding phase towards reformulating the business problem at hand.

6.1.3.6 Deployment

The deployment phase is concerned with the transfer of successful data-mining models to production. It is not confined to the integration of algorithms within platforms and database systems. Rather, it also includes the implementation of proper ways for presenting the information to the end users, including production of reports and dashboards.

Overall, CRISP-DM outlines the main activities that a data science team conducts when mining datasets, including IoT datasets. As already outlined, other data-mining processes (e.g., KDD, SEMMA) are quite similar and operate based on an iterative and phased approach as well.

6.1.4 IoT Data-Mining Challenges

The implementation of a data-mining process over IoT data is associated with several unique challenges, which render IoT data mining quite different from the mining of traditional batch datasets. These challenges stem from the properties of IoT data, which were introduced in Chapter 5. Indeed, the networked nature and high ingestion rates of IoT data streams introduce various challenges in the implementation of the above-listed data-mining steps [4]. Some of the most prominent challenges include:

- *Collecting and consolidating IoT data from diverse data sources:* Nontrivial IoT applications collect and process IoT data streams from multiple networked sources. This makes the data preparation phase very challenging, as diverse data streams featuring different formats and rates of ingestion need to be combined and processed. In several cases, there is also a need to unify the diverse semantics of the different data streams, as a means of boosting semantic data interoperability.
- *Executing mining functions on CPU-constrained devices:* Most IoT devices have quite limited computational capacity. This is the case with sensors, wearables, and RFID readers. As such, they have limitations when it comes to executing computationally intensive data-mining functions such as data preparation and machine learning models. This makes the development of a proper IoT data-mining system very challenging, as alternative solutions need to be evaluated and deployed. For example, data-mining functions could be moved to some edge server or IoT gateway with a proper amount of computational resources. In several cases, lightweight machine learning techniques (e.g., TinyML [18]) need to be designed and deployed.
- *Dealing with data velocity:* Real-time IoT applications need to process data streams with very high ingestion. Hence, it is not always possible to preprocess (e.g., filter) IoT data streams on-the-fly, given their very high velocity. Similarly, real-time knowledge extraction can become very challenging. These challenges are, to some extent, addressed by the IoT data streaming platforms that were presented in Chapter 5. Such platforms provide middleware services that facilitate large-scale analytics over IoT data streams.

However, there are also other ways to address data velocity challenges, such as the development of real-time data-mining algorithms that can be executed over streaming data.

- *Robustness challenging stemming from the properties of IoT data and devices:* Most IoT analytics applications consider the location of the IoT devices, as well as the timestamps of the data streams. Thus, changes in the location of devices and drifts in the network clocks can compromise the accuracy and the robustness of IoT data-mining applications.
- *Scalability challenges stemming from the number and distribution of IoT devices:* IoT data sources are characterized by significant variability, while being geographically and administratively distributed. These considerations make the IoT data-mining architectures challenging, especially in terms of their cost-effective scalability. Specifically, each new IoT device is likely to introduce revisions in the data-mining architecture, which is not the case with the scalability of traditional batch data sources and data stores.
- *Novel security, privacy, and data protection challenges:* IoT data-mining systems include more points of vulnerability such as the IoT devices and the networking infrastructure used to access them. As such, they are susceptible to a broader range of data breaches and security attacks. The latter are more extensively discussed in Chapter 7, including techniques for confronting them.

Overall, IoT data analytics systems are much more complex than conventional data-mining systems that operate over data at rest. The IoT analytics platforms and techniques of Chapter 5 provide a starting point for alleviating relevant challenges. Once an IoT data-mining system is in place, data can be collected and processed by machine learning techniques.

6.2 Data Mining and Machine Learning

6.2.1 Definitions and Common Data-Mining Tasks

6.2.1.1 Common Data-Mining Tasks

For the study and understanding of different machine learning methods, we introduce the following terms:

- *Instance:* An instance (also called an item or a record) is an example of an entity, described by several attributes. For example, a day can be described by temperature, humidity, and cloud status.
- *Attribute:* An attribute (also called a field) provides the means for measuring an aspect of the instance (e.g., a day's temperature).
- *Class (Label):* A class refers to a grouping of homogeneous instances (e.g., gold customers within a customer relationship management database).

Given the previous definitions, here are some very common tasks that data miners perform:

- *Classification:* This refers to the task of assigning an item to a class or in other words to the process of predicting the class of an item based on available datasets about similar items and their classes.
- *Clustering:* This refers to the process of grouping related data (instances) within a given dataset. In other words, clustering refers to finding clusters in data.
- *Associations:* This is the task of finding associated items (i.e., item attributes or conditions on data that occur frequently together).
- *Visualization:* This refers to the process of visualizing data in order to facilitate exploration of the datasets by humans, along with discovery of related knowledge.
- *Summarization:* This refers to the extraction of knowledge and attributes about a group of data.
- *Deviation detection:* This refers to the process of finding changes between given datasets and items within them.
- *Estimation:* This is the task of using datasets to predict a continuous value.
- *Link analysis:* This is about finding relationships between different datasets and their items.

6.2.1.2 Simple Data-Mining Problems

In order to facilitate the understanding of data-mining tasks, we will introduce a simple data-mining problem based on a publicly available dataset of the University of California (UC) Irvine Machine Learning Repository [5]. Specifically, we will use the "Echocardiogram" dataset, which contains echocardiogram data for a set of patients, along with other data that can be used for classifying whether patients will survive for at least 1 year after a heart attack [6]. An echocardiogram (or, simply, "echo") is a scan used to look at the heart and nearby blood vessels. It can facilitate the diagnosis and monitoring of certain heart conditions by checking the structure of the heart and surrounding blood vessels. It also facilitates analyzing and understanding how blood flows through them and assessing the pumping chambers of the heart. Nowadays, it is possible for cardiologists to employ connected devices for the echo scan, which produce digital data (i.e., digital echocardiography). The latter can be part of an IoT-based healthcare system.

Table 6.1 presents a snapshot of the dataset, including nine of the attributes that it contains. Each row of the dataset corresponds to data of a distinct patient. The nine attributes (features) can be interpreted as follows:

- *Survival:* This denotes the number of months that the patient survived or has survived (in case the patient is still alive). Because all the patients had their heart attacks at different times, it is possible that some patients have survived

Table 6.1 A Snapshot of the Echocardiogram Dataset of the UC Irvine Machine Learning Repository

Survival	*Still-Alive*	*Age-at-Heart-Attack*	*Pericardial Effusion*	*Fractional Shorening*	*Epss*	*Lvdd*	*Wall-Motion Index*	*Alive at One*
11	0	71	0	0.26	9	4.6	1	0
19	0	72	0	0.38	6	4.1	1.7	0
16	0	55	0	0.26	4	3.42	1	0
57	0	60	0	0.253	12.062	4.603	1.45	0
19	1	57	0	0.16	22	5.75	2.25	0
26	0	68	0	0.26	5	4.31	1	0
13	0	62	0	0.23	31	5.43	1.875	0
50	0	60	0	0.33	8	5.25	1	0
19	0	46	0	0.34	0	5.09	1.14	0
25	0	54	0	0.14	13	4.49	1.19	0
10	1	77	0	0.13	16	4.23	1.8	1
52	0	62	1	0.45	9	3.6	1.14	0
52	0	73	0	0.33	6	4	1	0
44	0	60	0	0.15	10	3.73	1	0
0.5	1	62	0	0.12	23	5.8	2.33	1
24	0	55	1	0.25	12.063	4.29	1	0
0.5	1	69	1	0.26	11	4.65	1.64	1
0.5	1	65.529	1	0.07	20	5.2	2	1
22	1	66	0	0.09	17	5.819	1.333	0
1	1	66	1	0.22	15	5.4	2.25	1

less than 1 year, but they are still alive. Such patients cannot be used for the prediction task outlined later.

- *Still-alive:* This is a binary variable that denotes whether the patient is still alive, that is, it has a value of zero (0) if the patient is dead at the end of survival period or one (1) if the patient is still alive.
- *Age-at-heart-attack:* This denotes that age of the patient in years when the heart attack occurred.
- *Pericardial-effusion:* This is another binary variable that denotes whether pericardial effusion is fluid around the heart (i.e., one (1)) or not (i.e., zero (0)).
- *Fractional-shortening:* This is a measure of contractility around the heart. Lower numbers are increasingly abnormal.
- *Epss:* This is the E-point septal separation, which is another measure of contractility. Larger numbers are increasingly abnormal.
- *Lvdd:* This attribute denotes the left ventricular end-diastolic dimension. It is a measure of the size of the heart at end-diastole. Large hearts tend to be sick hearts.

- *Wall-motion-index:* This corresponds to the division of the wall-motion-score (i.e., a measure of how the segments of the left ventricle are moving) with the number of segments seen. Usually, 12 to 13 segments are seen in an echocardiogram.
- *Alive-at-One:* This is Boolean value that is derived from the first two attributes. Zero means that the patient was either dead after 1 year or had been followed for less than 1 year. One (1) means that the patient was alive at 1 year.

This dataset is suitable for a simple classification problem. The latter lies in training a machine learning program to tell whether the patient will survive for at least 1 year or not. The dataset can be used for training the program. The latter will be then able to tell the value of the last attribute based on the values of earlier attributes for a new patient. In practice, the classification problem could be simply expressed as follows: Given information from the echocardiogram of a patient who has had a heart attack, develop a machine learning classifier that would be able to tell whether the patient will survive for the next 12 months.

This dataset was mainly composed of numeric values, including ordinal values (i.e., values in the attributes that are ordered). However, there are also datasets that comprise nominal values (i.e., values without any set order). Typical examples of such values can be found in attributes with yes and no values. A relevant dataset is presented in Table 6.2, which presents a snapshot of a medical dataset that enables the diagnosis of acute inflammations of urinary bladder [7]. The dataset can be also downloaded from the UC Irvine Machine Learning Repository and comprises the following attributes:

- The temperature of the patient: a value between 35°C and 42°C;
- The occurrence of nausea: yes if present and no if not present;
- Lumbar pain: yes if present and no if not present;
- Urine pushing: continuous need for urination, yes if present and no if not present;
- Micturition pains: yes if present and no if not present;
- Burning of urethra (itch, swelling of urethra outlet): yes if present and no if not present;
- Inflammation of urinary bladder: yes if present and no if not present.

Using this dataset, it is possible to train a machine learning program to classify a patient as having acute inflammation of the urinary bladder or not. Specifically, this machine learning system would be able to classify a patient as having or not having acute inflammation of the urinary bladder, given the six first attributes of this list.

In general, the process of building a classifier involves using a subset of known results, which is used as a training set, to build a model. This model is evaluated against a testing set, which is typically used to calculate the error rate of the

Table 6.2 Snapshot of the Dataset for the Detection of the Acute Inflammation of the Urinary Bladder

Temperature	*Nausea*	*Lumbar Pain*	*Urine Pushing*	*Micturition Pains*	*Burning of Urethra*	*Inflammation*
35.5	No	Yes	No	No	No	No
35.9	No	No	Yes	Yes	Yes	Yes
35.9	No	Yes	No	No	No	No
36.0	No	No	Yes	Yes	Yes	Yes
36.0	No	Yes	No	No	No	No
36.0	No	Yes	No	No	No	No
36.2	No	No	Yes	Yes	Yes	Yes
36.2	No	Yes	No	No	No	No
36.3	No	No	Yes	Yes	Yes	Yes
36.6	No	No	Yes	Yes	Yes	Yes
36.6	No	No	Yes	Yes	Yes	Yes
36.6	No	Yes	No	No	No	No
36.6	No	Yes	No	No	No	No
36.7	No	No	Yes	Yes	Yes	Yes
36.7	No	Yes	No	No	No	No
36.7	No	Yes	No	No	No	No
36.8	No	No	Yes	Yes	Yes	Yes
36.8	No	No	Yes	Yes	Yes	Yes
36.9	No	No	Yes	Yes	Yes	Yes
36.9	No	Yes	No	No	No	No
37.0	No	No	Yes	Yes	No	Yes
37.0	No	No	Yes	Yes	No	Yes
37.0	No	Yes	No	No	No	No
37.0	No	No	Yes	Yes	Yes	Yes
37.0	No	No	Yes	Yes	Yes	Yes
37.0	No	Yes	Yes	Yes	Yes	Yes
37.0	No	No	Yes	Yes	Yes	Yes
37.0	No	No	Yes	No	No	Yes
37.1	No	Yes	No	No	No	No

classifier. The latter process involves a comparison of the classifications produced by the classifier with the known classes in the testing sets. In line with the CRISP-DM data-mining process, the evaluation may lead to changes and reconsiderations about the model, including the development of alternative models.

6.2.1.3 Building the Simplest Classifier: The 1R Learner

There are many machine learning models that could be used for developing a classifier for the Acute Inflammation of Urinary Bladder prediction. The simplest one is the 1R Learner, an empirical learning method that takes as input a set of examples that comprise several attributes (like the ones in Table 6.2) and a class (i.e., "Inflammation" in this case). Accordingly, 1R infers a rule that predicts the class

given the values of the attributes. The 1R works by choosing the most informative single attribute (i.e., it bases the classification the rule on this attribute alone) [8]. In practical terms, 1R uses the most characteristic attribute for the outcome at hand towards performing the classification.

Identifying the attribute that is the best predictor of the inflammation outcome is quite straightforward. It is all about identifying how different set of classification rules based on each one of the attributes perform on the training data. Table 6.3 compares the accuracy (or the errors) of three of the attributes (i.e., urine pushing, burning of urethra, lumbar pain) on 120 of the known patient cases of the training dataset.

The outcomes listed in Table 6.3 suggest that "Urine pushing" is a better indicator of acute inflammation of the urinary bladder, when compared to the other two symptoms. The 1R learner can be represented as a one-level decision tree that tests the one feature to classify an instance to one of the available classes.

6.2.1.4 Building Decision Trees

A 1R learner decision tree cannot typically classify all the cases of the training set correctly, as it is likely that no attribute can be a good predictor of the target attribute at all uses. To alleviate this limitation, one can build a multilevel decision tree that uses a combination of attributes to predict whether the patient has acute inflammation based on his or her symptoms. The development of nontrivial machine learning models like a decision tree is usually supported by appropriate tools for data science and machine learning, including programming languages like Python, Java, and R, as well as visual environment for data analytics. In our case, we used the popular KNIME (Konstanz Information Miner) platform, which provides a free, open-source environment for data analytics, reporting, and integration [9]. KNIME integrates various components for machine learning and data mining based on a modular and flexible data pipelining concept. According to this concept, each data-mining task is represented as a data pipeline or data workflow. The pipeline specifies how data flows from data sources to machine learning outcomes and their visualizations that are presented to end users. KNIME provides very good support for all phases of the CRISP-DM process, including data preparation, modeling, evaluation, and visualization tasks.

Table 6.3 Comparison of Classification Error Rates for Different Attributes

Attribute	*Classification Rules*	*Errors*
Urine pushing	Urine pushing = No ➔ Inflammation = No	21/120
	Urine pushing = Yes ➔ Inflammation = Yes	
Burning of urethra	Burning of urethra =No➔ Inflammation = No	51/120
	Burning of urethra = Yes ➔ Inflammation = Yes	
Lumbar pain	Lumbar pain = No ➔ Inflammation = Yes	29/120
	Lumbar pain = Yes ➔ Inflammation = No	

Figure 6.2 depicts a KNIME workflow that builds and evaluates a decision tree for the acute inflammation classification problem. The workflow comprises the following modules (i.e., KNIME nodes):

- *CSV Reader:* This is a KNIME node that reads data from a CSV file.
- *Column Filter:* This node filters out columns of the original datasets, which are not relevant for the development of the decision tree. The outcome of the node is a transformed (i.e., reduced) version of the original dataset.
- *Partitioning:* This KNIME module partitions the available data set to two parts: a training dataset comprising 80% of the records and a test dataset that comprises the rest 20%. The training dataset will be used to train the decision tree, while the testing dataset will be used to evaluate its classification ability. Obviously, the training and testing datasets should be different. The ratio of training and testing records is configurable. The 80%-20% is a rule of thumb that is commonly applied in machine learning activities.
- *Decision Tree Learner:* Using the training dataset, this node creates a decision tree for the acute inflammation classification problem. The tree is illustrated in Figure 6.3 and includes four levels that allow it to correctly classify all instances of the training dataset. The Decision Tree Learner node can be configured with respect to its operation. Information about its configuration can be found in the node's documentation in KNIME [10]. In principle, a learned decision tree prioritizes the attributes that are good predictors of the outcome, as a means of keeping the depth of the tree as short as possible. This is key to achieving the faster possible performance. In this context, the first attribute tested by the tree is the "Urine pushing," which is the best possible predictor of the outcome, as earlier illustrated in the 1R tree presentation. The use of this attribute enables the fast classification of all the cases without pushing symptoms. Indeed, based on the training dataset, the lack of urine pushing implies automatically a lack of inflammation.

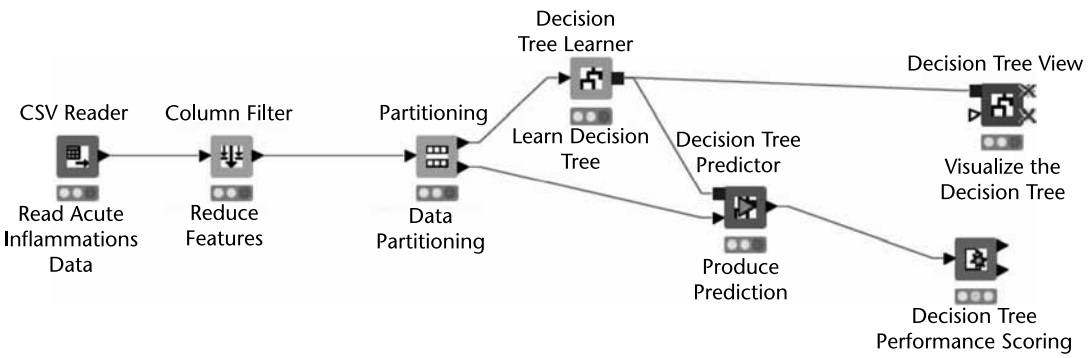

Figure 6.2 Producing and executing a decision tree classifier for the acute inflammations dataset as a KNIME workflow.

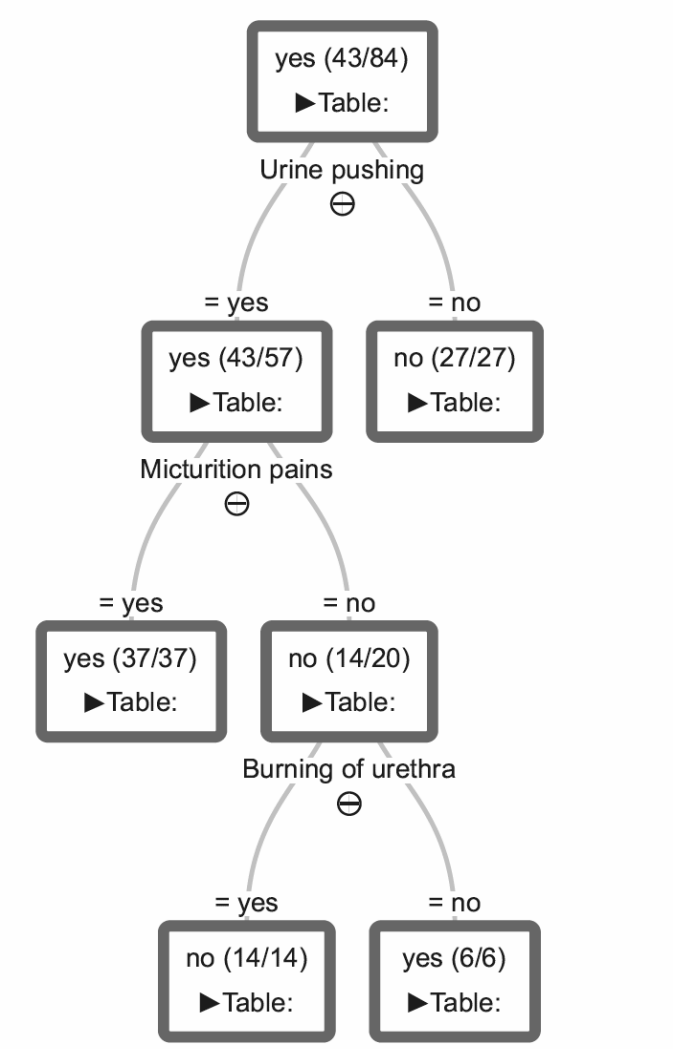

Figure 6.3 A decision tree for the acute inflammation classification problem.

- *Decision Tree View:* This node visualizes the learned decision tree as shown in Figure 6.3. Hence, its outcome is directly linked to the Decision Tree Learner node.
- *Decision Tree Predictor:* This node applies the decision tree on the test dataset. It therefore takes as input the test partition of the Partitioning node.
- *Scorer:* The Scorer node produces different metrics that are used to evaluate the accuracy of the predictor. For example, it calculates the popular precision and recall measures [11]. Precision (or positive predictive value) is the fraction of correctly classified instances among the retrieved instances, while recall (or sensitivity) is the fraction of the total amount of relevant instances that were retrieved. In the case of the acute inflammation dataset, the trained

decision tree achieves perfect precision (i.e., no false-positive) and perfect recall (i.e., no false-negative) on the test dataset.

Figure 6.4 applies a similar workflow towards building and evaluating a decision tree model for the echocardiogram problem (i.e., the prediction of whether patients will be alive for 1 year, based on the parameters of their echo scan). The workflow is similar, yet it applies different preprocessing functions in order to prepare the dataset to be fed into the Decision Tree Learner.

The decision tree that was built with the KNIME tools exploited several attributes for the classification. Intuitively, this seems a better idea that using just one attribute. A quite simple algorithm that considers all the available attributes is based on a Bayes theorem and is conveniently called the Naïve Bayes Learner. The idea relies on identifying the impact of each one of the features on the outcome, as a means of weighting them accordingly. While this may sound like a good idea, it is based on the rather unrealistic assumption that all features are independent from each other. This assumption lies behind the fact that all attributes contribute equally and independently on the outcome. However, the Naïve Bayes Learner is a popular algorithm for simple data-mining problems.

Simple algorithms such as 1R, Naïve Bayes, and Decision Trees work surprisingly well for simple problems and standard machine learning datasets. However, for complex problems combinations of different algorithms are usually used in the industry. Next, we present more techniques for solving a classification problem, including regression, decision trees, Bayesian approaches, and neural networks. We also introduce the different types of machine learning, including deep learning and reinforcement learning.

6.2.2 Classification of Machine Learning Techniques

Independently of the specific data-mining task that they address, machine learning algorithms can be classified as follows.

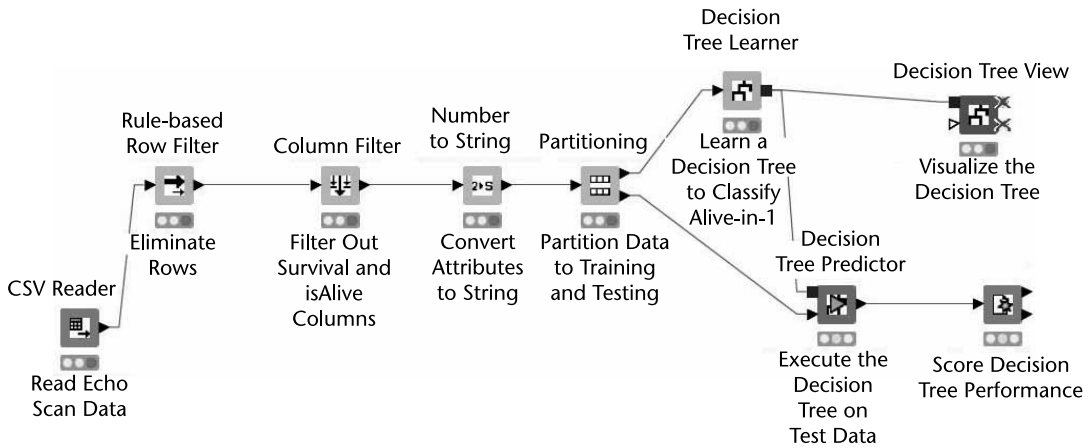

Figure 6.4 Producing and executing a decision tree classifier for the echocardiogram dataset as a KNIME workflow.

- *Supervised learning:* In supervised learning, a property (label) is available for a certain dataset (i.e., the training set). This same property is missing for new instances and hence needs to be produced based on (supervised) machine learning. Decision trees, Naïve Bayesian classification, and support vector machines are some popular supervised learning techniques. The problems that were introduced previously were based on supervised learning, as labeled data with information about the outcomes of the classification were available.
- *Unsupervised learning:* Unsupervised learning is about discovering implicit relationships within an unlabeled dataset [12]. Hence, in unsupervised learning, items are not preassigned and no training datasets are used. Clustering methods provide a prominent example of unsupervised learning. In practice, unsupervised learning methods are used in the absence of labeled data. In such cases, unsupervised learning approaches can help to identify abnormalities such as instances that deviate from the usual behavior of the majority.
- *Reinforcement learning:* Reinforcement learning is usually used for more complex problems such as recognition of complex patterns. It involves taking actions in order to maximize cumulative rewards [13]. Reinforcement learning is typically formulated in a Markov decision process (MDP) environment and entails dynamic programming techniques. It is used for applications involving game theory, operations research, and multi-agent systems. In general, reinforcement learning is a more advanced form of machine learning, which is employed in several AI applications, such as the famous Google's AlphaGO AI engine, which in March 2016 managed to beat one of the world's best experts in the "GO" game. Most data scientists work with supervised and unsupervised learning, rather than with reinforcement learning. However, reinforcement learning is gaining momentum in IoT applications such as robots and autonomous driving, where moving objects are incentivized to learn the proper behavior (e.g., collision avoidance, safe wandering, safe driving).

In recent years, we have also witnessed an expanded use of deep learning. Deep learning is a special type of machine learning that leverages algorithms inspired by the structure and function of the brain. Specifically, deep learning leverages artificial neural networks, notably neural networks that have many (deep) layers that enable learning. It can be both supervised and unsupervised, depending on whether labeled data are used. Deep learning is currently used in many IoT applications such as autonomous driving, automated guided vehicles, and complex image classification. Its growing momentum is largely due to that as more data become available, deep learning improves its accuracy much better than other machine learning algorithms (e.g., decision trees and Bayesian algorithms). In an era where data are generated in an exponential pace, this property is very important. IoT systems and applications are among the main contributors to this ever-increasing data generation, which is the reason why deep learning is very widely deployed in the scope of IoT applications.

6.3 Popular Machine Learning Models and Techniques

6.3.1 Decision Trees

Decision trees were previously introduced. They are one of the most popular, intuitive, and easy-to-understand classification methods. From a technical viewpoint, they are decision support tools that use a tree-like graph or model of decisions. They operate through evaluating chance-event outcomes, resource costs, and utility of decisions in each step. From a business viewpoint, decision trees represent the minimum number of yes or no questions that one must ask in order to assess the probability of making a correct decision. Hence, they provide a structured and systematic way to arrive at a logical conclusion. A simpler and more intuitive way for perceiving a decision tree-based classifier is to view it as a series of if-then-else statements.

6.3.2 Least Squares Regression (LSR)

Another classification method is the LSR. It is a method of performing linear regression (i.e., fitting a straight line through a set of points in an optimal way). Optimality is specified in terms of the vertical distances between the point(s) and the line. In particular, the distances of the various points are added and the line that gives the smallest possible sum provides the fitted line.

The term "linear" in the title of the LSR denotes the kind of model used to fit the data. At the same time, the term "least squares" denotes the error metric that is minimized in order to lead to the optimal fit. In particular, "least squares" refers to summing and them minimizing the squares of the distances of the given points from the line.

6.3.3 Naïve Bayes Classification

Naïve Bayes classification refers to a family of simple probabilistic classifiers based on the Bayes theorem. The Bayes theorem is used to calculate the posterior probability for a class c, given the probability of a known event x (i.e., $P(c \mid x)$, based on the likelihood of the event x for class c, the prior probability for an event being in class c, and the predictor prior probability of the known event X ($P(x)$). This relationship is straightforward, assuming independence between the features and events x, c. As already outlined, this assumption is unrealistic in practice. However, there are cases where some strong independence between the events can be assumed.

Classifiers based on the "naïve" versions of the Bayesian theorem are very easy to implement. It is quite impressive that is several occasions they give decent results. Their operation is based on the calculation of the posterior probability for an event or item being in class c given an event x, which is observed in the available dataset.

Despite their simplicity, naïve Bayesian classifiers have several applications such as:

- Marking an email as spam or not spam, given the previous (known) classification of other emails as spam or not spam;
- Classify a news article about technology, politics, or sports, based on some of its attributes (e.g., words) and given the known classification of other articles in the available datasets;
- Check a piece of text expressing positive emotions or negative emotions, which is known as sentiment analysis;
- Classifying faces in some known categories (face recognition) based on some facial features.

6.3.4 Logistic Regression

Logistic regression provides a powerful statistical way of modeling a binomial outcome with one or more explanatory variables. It measures the relationship between the categorical dependent variable and one or more independent variables. Accordingly, it estimates probabilities using a logistic function, which is the cumulative logistic distribution. Logistic regression provides a probability fashion for the classification based on a logistic rather than a linear model. The logistic model can provide a much more accurate probability distribution than a linear model. Real-world applications, where logistic regression is applied, include credit scoring, measuring success rates of marketing campaigns, predicting revenues of a certain product, and predicting a specific weather characteristic within a given day and more.

6.3.5 Support Vector Machines (SVN)

The SVN is a popular binary classification algorithm. Given a set of points of two types in N dimensional place, SVM generates a (N – 1) dimensional hyperplane to separate those points into two groups. SVM produces a straight line, which separates two points into two types situated as far as possible from all those points.

SVN classification is used in various practical examples, including display advertising, human splice site recognition, image-based gender detection, and large-scale image classification.

6.3.6 Ensemble Methods

Ensemble methods refer to a class of learning algorithms that construct a set of classifiers and then classify new data points by taking a weighted vote of their predictions. Originally, ensemble methods used Bayesian averaging. However, more recent algorithms were expanded to include error-correcting output coding, bagging, boosting, and other methods. The main advantages of sampling methods are that:

- They average out biases and hence provide more balanced results.
- They reduce variance due to diversification (e.g., like in a stocks portfolio).

- They are unlikely to over-fit, since they combine predictions from non-over-fitting models (e.g., average, weighted average, logistic regression).

6.3.7 Clustering

Clustering represents a whole range of methods for grouping a set of objects into various categories in a way that objects in the same group (cluster) are more like each other than to those in other groups. Prominent examples of clustering algorithms include centroid-based algorithms, connectivity-based algorithms, density-based algorithms, probabilistic algorithms, dimensionality reduction algorithms, and neural networks. Neural networks comprise also a class of deep learning algorithms, which are nowadays very popular in terms of their ability to identify complex patterns in multimedia (IoT) datasets.

Clustering algorithms fall in the class of unsupervised learning algorithms. This means that the classification of instances in various classes is not based on the exploitation of existing labeled datasets that are used as training datasets. Rather, clustering algorithms find a natural grouping of instances given unlabeled data as shown in the figure where instances are groups in three areas in space.

Neural networks are a popular clustering technique, which can be used to select more complex regions as part of the clustering process. Furthermore, neural networks can yield more accurate results. On the down side, they can overfit the data, which means that they could lead to the identification of patterns in random noise.

Different clustering algorithms have been devised to address a variety of different (data mining) cases, including:

- Cases involving numeric or symbolic data;
- Deterministic and probabilistic data modeling cases;
- Hierarchical and flat cases;
- Top-down and bottom-up cases.

The evaluation of the effectiveness of the clustering algorithms can be performed based on different techniques, including manual inspection, benchmarking on existing labels, and cluster quality measures based on different metrics (e.g., distance measures, high similarity within a cluster, low across clusters).

K-means is one of the most popular clustering algorithms. It works with numeric data only and operates as follows:

- It picks a number (K) of cluster centers (at random).
- It assigns every item to its nearest cluster center (e.g., using Euclidean distance).
- It moves each cluster center to the mean of its assigned items.

The last two steps are repeated until convergence is achieved. Convergence is achieved when change in cluster assignments is less than a threshold.

6.3.8 Principal Component Analysis (PCA)

PCA is one more statistical analysis technique that is used in machine learning contexts. It is a statistical procedure that uses an orthogonal transformation, which converts a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. The linearly uncorrelated variables are called principal components (or, sometimes, principal modes of variation). The number of principal components is less than or equal to the smaller of the number of original variables or the number of observations. Popular applications of PCA can be found in interest rate derivatives portfolios and neuroscience.

6.3.9 Independent Component Analysis (ICA)

ICA is another statistical technique for revealing hidden factors that underlie sets of random variables, measurements, or signals. It defines a generative model for the observed multivariate data, which is typically given as a large database of samples. In the model, the data variables are assumed to be linear mixtures of some unknown latent variables, and the mixing system is also unknown. Latent variables are assumed to be non-Gaussian and mutually independent, and they are called independent components of the observed data. ICA is a special case of blind source separation (BSS), which is very commonly used in problems involving audio signals. A very popular problem is the "cocktail party problem," which refers to the task of separating different conversations in the scope of a noisy room where different groups of people speak simultaneously.

ICA is related to PCA as it transforms data into independent components. Nevertheless, it is considered much more powerful than PCA for several problems. ICA is used in many different applications including digital images classification, document databases, economic indicators identification and psychometric measurements.

A wider range of machine learning techniques, including deep learning and reinforcement learning algorithms can be found in the large number of relevant surveys (e.g., [14–16]).

6.4 Models Evaluation

6.4.1 Overview

In the presence of many different algorithms for mining data, it is important to be able to select the method that is the most effective for the problem at hand. In principle, no model is uniformly the best. When mining data, scientists are expected to evaluate and test many different methods in order to identify the optimal one. The comparison of different methods can be performed against different dimensions depending on the business requirements. Some of such dimensions include:

- The speed of training of the machine learning model;
- The speed of model application;

- The noise tolerance of the method;
- The extent to which it is intuitive, understandable, and explanatory, which all highly depend on the semantics of the selected model.

In real-life problems, hybrid integrated models are very commonly used as they yield better results than any single method alone.

6.4.2 Classification Techniques Evaluation

To evaluate different classification techniques (e.g., decision trees, linear regression), one needs to assess their effectiveness in terms of their ability to classify new instances. To this end, the error on the training data is not a good indicator of performance on future data. This is quite reasonable given that the new data will not be the same as the training data. Furthermore, the different methods must alleviate the overfitting problem. Overfitting refers to fitting the model too precisely on the training data, which usually leads to poor results on new data. Hence, other data should be used for the evaluation. Earlier, we outlined the important of data partitioning, which typically take places based on an 80%-20% rule [17]. Overall, a test dataset that is totally disjoint from the training dataset needs to be used for the evaluation of a classification method.

The evaluation of different classification methods and measures can be based on the following metrics:

- *Classification accuracy:* This is the ability of a classifier to classify new instances correctly.
- *Total cost/benefit:* This occurs especially when different errors involve different costs. Cost/benefit is a business indicator as errors are in many cases associated with business costs or business risks.
- *Lift and receiver operating characteristic (ROC) curves:* These are graphical plots that illustrate the performance of a binary classifier system as its discrimination threshold is varied.
- *The error in numeric predictions:* These occur in cases where the prediction concerns a numeric value rather than a class.

6.4.3 Classifier Error Rate

The use of the classifier error rate is a very popular method for ranking the performance and efficiency of classification methods. Its calculation is based on the following simple and straightforward principles:

- When an instance's class is predicted correctly, the classifier is successful.
- When an instance's class is predicted incorrectly, the classifier is erroneous.
- The error rate is defined as the proportion of errors made over the whole set of instances.

As already outlined, it is not credible to calculate the error rate on the training set, as this will give way too optimistic results. Rather, additional data are required for computing the error rate.

6.5 Data-Mining Models for IoT

6.5.1 Overview of IoT Data-Mining Models

All the previously presented data-mining models and techniques are applicable to all types of datasets, including IoT datasets, other streaming datasets (e.g., Twitter data), and traditional batch datasets. In the case of IoT applications, data-mining techniques are fed from data stemming from IoT data sources such as sensors, robots, CPS, wearables, and other internet-connected devices. This requires the deployment of machine learning models over the IoT analytics platforms described in Chapter 5. Some of the platforms discussed in this chapter include readily available libraries and toolkits for building and executing machine learning models.

The deployment of machine learning models in IoT environments can be performed in the following prominent ways:

- *Multilayer data-mining models:* These entail IoT pipelines involving data collection, data processing, event processing, and data-mining services.
- *Distributed data-mining models:* These are used to solve problems associated with the storage of IoT data in different locations.

These models address the peculiar challenges of IoT data mining, which are as follows:

- *IoT data are real-time:* This introduces timing and streaming challenges.
- *IoT data are provided based on an uninterrupted data flow:* This should be addressed at the level of IoT nodes.
- *IoT data can be potentially unlimited:* This makes it challenging to store and process them in memory.
- *IoT applications are dynamic:* This asks for constructing data models and machine learning models that are dynamic and adaptive, rather than being static and unchanged during the lifetime of the IoT application.
- *Several IoT applications involve actuation and use the data to drive decisions and provide intelligent feedback to other systems:* This is another factor that differentiates IoT from other types of data-intensive systems.

6.5.2 The Multilayer Data-Mining Model

The multilayer data model for IoT data mining, incudes a set of layered components as follows:

- *Data collection layer:* This is in charge of IoT data collection and acquisition based on appropriate interfaces to sensors and other IoT devices.
- *Data management layer:* This undertakes the storage of IoT data in a centralized or distributed database (e.g., SQL or no-SQL database). At this layer, data could also be structured and persisted in a data warehouse.
- *Event processing layer:* This undertakes to process (e.g., filter) the IoT data towards producing events.
- *Event mining layer:* This deploys data mining and machine learning techniques in order to extract knowledge from the IoT data.

Most IoT platforms employ this multilayer data model (or variations of it) in order to enable analytics based on IoT data.

Each of the layers of the multilayer data model provides some added value functionalities as follows:

- The data collection layer deals with energy-saving features, alleviates misreading, provides the means for repeatable reads, ensures fault tolerance and basic data filtering close to the source, and undertakes networked communications with the systems supplying the data.
- The data management layer manages collected data based on centralized or distributed database or even a data warehouse. It undertakes data abstraction and compression.
- The event-processing layer is used effectively to analyze events and perform query analysis based on the events.
- The data-mining service layer deals with data mining and production of new knowledge from the IoT datasets.

With reference to IoT architectures and topologies illustrated in earlier chapters, machine learning over IoT data can be deployed either at the edge or at the cloud levels of an IoT architecture. Edge deployments should typically comprise high-performance, low-latency machine learning models that are executed close to the field. However, machine learning models that do not have real-time requirements can be executed on the cloud. Such models can benefit from the collection of large volumes of IoT data in the cloud, towards building accurate models leveraging on techniques such as deep learning.

6.6 Summary

This chapter has presented processes and techniques for mining IoT data. The process of mining IoT datasets is no different from the classical machine learning techniques that are used in mining data in conventional databases. It involves using data to train machine learning models, which are accordingly tested and validated against a test dataset. However, the use of machine learning techniques in IoT applications requires the acquisition of data from IoT data sources such as sensors,

CPS, and other internet-connected devices. In several cases, IoT applications ask for fast and high-performance machine learning algorithms, given their need to deal with streaming data. Likewise, IoT platforms for streaming analytics are, in several cases, needed as the baseline infrastructures that enable the collection and preprocessing of the data, which are fed to machine learning algorithms.

IoT data scientists and IoT solution developers are nowadays offered with a host of machine learning algorithms that can help them to extract insights from their IoT data. Moreover, they are also provided with access to a rich set of data science tools that boost their productivity when performing IoT analytics tasks. Furthermore, IoT solution providers can greatly benefit from tools and techniques for special types of machine learning (e.g., deep learning) that are particularly suitable for leading-edge IoT applications such as autonomous driving. Overall, developers and deployers of the IoT solutions must possess data science expertise, as a key prerequisite for mining IoT datasets in optimal way and getting the most out of machine learning and AI. In this direction, it is very important to ensure the security and trustworthiness of IoT-based data-mining systems that comprise more points of vulnerabilities than conventional data-mining system. To this end, there is a need for securing nontrivial IoT systems. This is the focus of Chapter 7.

References

[1] Mahdavinejad, M. S., et al., "Machine Learning for Internet of Things Data Analysis: A Survey," *Digital Communications and Networks*, Vol. 4, No. 3, 2018, pp. 161–175.

[2] Azevedo, A., and M. F. Santos, "KDD, SEMMA and CRISP-DM: A Parallel Overview," *Proc. of IADIS European Conference on Data Mining*, 2008, pp. 182–185.

[3] Shearer, C., "The CRISP-DM Model: The New Blueprint for Data Mining," *J. Data Warehousing*, Vol. 5, 2000, pp. 13–22.

[4] Tsai, C. -W., et al., "Data Mining for Internet of Things: A Survey," *IEEE Communications Surveys and Tutorials*, Vol. 16, No. 1, First Quarter 2014.

[5] Dua, D., and C. Graff, *UCI Machine Learning Repository*, Irvine, CA: University of California, School of Information and Computer Science, 2019, http://archive.ics.uci.edu/ml.

[6] Salzberg, S., *Exemplar-Based Learning: Theory and Implementation*, Technical Report TR-10-88, Harvard University, Center for Research in Computing Technology, Aiken Computation Laboratory Cambridge, MA, 1988.

[7] Czerniak, J., and H. Zarzycki, "Application of Rough Sets in the Presumptive Diagnosis of Urinary System Diseases, Artificial Intelligence and Security in Computing Systems," *ACS 2002 9th International Conference Proceedings*, 2003, pp. 41–51.

[8] Holte, R. C., "Very Simple Classification Rules Perform Well on Most Commonly Used Datasets," *Machine Learning*, Vol. 11, 1993, pp. 63–90.

[9] Feltrin, L., "KNIME an Open Source Solution for Predictive Analytics in the Geosciences [Software and Data Sets]," *IEEE Geoscience and Remote Sensing Magazine*, Vol. 3, 2015, pp. 28–38.

[10] Shafer, J. C., R. Agrawal, and M. Mehta, "SPRINT: A Scalable Parallel Classifier for Data Mining," *Proc. of 22th International Conference on Very Large Data Bases (VLDB '96)*, San Francisco, CA, 1996, pp. 544–555.

[11] Powers, D. M., "Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation," *J. Mach. Learn. Technol.*, Vol. 2, 2011, pp. 2229–3981.

[12] Vachkov, G., and H. Ishihara, "Unsupervised Learning Algorithms for Comparison and Analysis of Images," *2008 IEEE International Conference on Mechatronics and Automation*, Takamatsu, 2008, pp. 415–420.

[13] Qiang, W., and Z. Zhongli, "Reinforcement Learning Model, Algorithms and Its Application," *2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC)*, Jilin, 2011, pp. 1143–1146.

[14] Kour, H., and N. Gondhi, "Machine Learning Techniques: A Survey," in J. Raj, A. Bashar, and S. Ramson, (eds.), *Innovative Data Communication Technologies and Application (ICIDCA 2019), Lecture Notes on Data Engineering and Communications Technologies*, Vol. 46, 2020.

[15] Obulesu, O., M. Mahendra and M. ThrilokReddy, "Machine Learning Techniques and Tools: A Survey," *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*, Coimbatore, 2018, pp. 605–611.

[16] Shanthamallu, U. S., et al., "A Brief Survey of Machine Learning Methods and Their Sensor and IoT Applications," *2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA)*, Larnaca, 2017, pp. 1–8.

[17] Inyaem, U., "Construction Model Using Machine Learning Techniques for the Prediction of Rice Produce for Farmers," *2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)*, Chongqing, 2018, pp. 870–874.

[18] Sanchez-Iborra, R., and A. F. Skarmeta, "TinyML-Enabled Frugal Smart Objects: Challenges and Opportunities," *IEEE Circuits and Systems Magazine,* Vol. 20, No. 3, 2020, pp. 4–18.

CHAPTER 7

IoT Security and Privacy

Deploying an IoT system in production requires strong security. IoT security therefore is a prerequisite for the wider adoption and acceptance of the IoT paradigm, as poor security can have severe implications for the deployer and operator of the IoT system. In some cases, poor security could have life-threatening implications. As a prominent example, compromising the security of self-driving cars could put the passengers' lives at risk. From a technical perspective, IoT security is particularly challenging, given that it involves security mechanisms at multiple levels, including security mechanisms for devices, networks, cloud computing infrastructures, and IoT applications.

IoT systems must also operate in a privacy-friendly way, especially in cases where personal data are collected and processed. This is the case with IoT-based healthcare applications, which deal with sensitive data of the patients.

This chapter presents the main IoT security and privacy challenges. It illustrates why IoT systems are different from other types of cyber infrastructures. Moreover, it introduces a rich set of security and privacy solutions that serve different purposes and address various threats. In the context of these solutions, this chapter discusses how emerging technologies (e.g., machine learning and blockchain technologies) can be used to implement novel IoT security mechanisms.

7.1 Understanding IoT Security

7.1.1 The IoT Security Jargon

The development and deployment of IoT security systems require a specification of components that comprise the system, the structuring principles that drive the integration of these components, and how they could be attacked by an adversary. In this direction, there is a need for understanding the following IoT security concepts:

- *Asset:* This refers to a component of the IoT system that supports its operation. Typical examples of assets are the data, the devices, the IoT software, and the IoT services that comprise an IoT system. In general, an IoT system can be broken down into various assets. In most cases, attackers compromise the operation of an IoT system through attacking one or more assets of the system.

- *Threat:* A threat in an IoT security context refers to a possible danger that is associated with the security of the system. Specifically, it is a danger that stems from one or more vulnerabilities of the system. The exploitation of these vulnerabilities can lead to a security breach on the system,
- *Vulnerability:* This refers to a weakness of the IoT system, which can be exploited by a malicious party or adversary. Attackers exploit vulnerabilities in order to perform unauthorized actions in an IoT system. In this direction, they usually leverage some adversarial technique to connect to the system and to take advantage of the identified vulnerability.
- *Risk:* A risk represents the impacts of a security breach or attack, which could take place due to some threat or vulnerability. It refers to the impacts on the organizations and stakeholders that are entailed in the operation and use of an IoT system.
- *Attack:* Refers to an attempt to exploit, expose, alter, disable, destroy, or compromise an asset. It may also entail gaining unauthorized access to an IoT asset. There is a very broad range of IoT attacks, which partly explains the complexity of IoT security.

These concepts facilitate the understanding of the structure of IoT security systems and processes. Overall, IoT security concerns the protection of IoT systems and their assets from attacks. In this direction, IoT security mechanisms attempt to minimize potential threats and vulnerabilities. Moreover, they focus on identifying and mitigating potential risks. The overall goal is to avoid security breaches and to protect IoT systems from disruption or misuse of the services they provide.

IoT security is generally more complex and challenging than security for conventional information systems. This is largely due to that nontrivial IoT systems comprise multiple assets and components, such as devices, networks, clouds, and IoT applications. Figure 7.1 illustrates that end-to-end IoT security entails security multiple components at different levels, including devices, networks, cloud computing infrastructures, and various applications. This is further elaborated next, where we present the drivers behind security modern IoT systems.

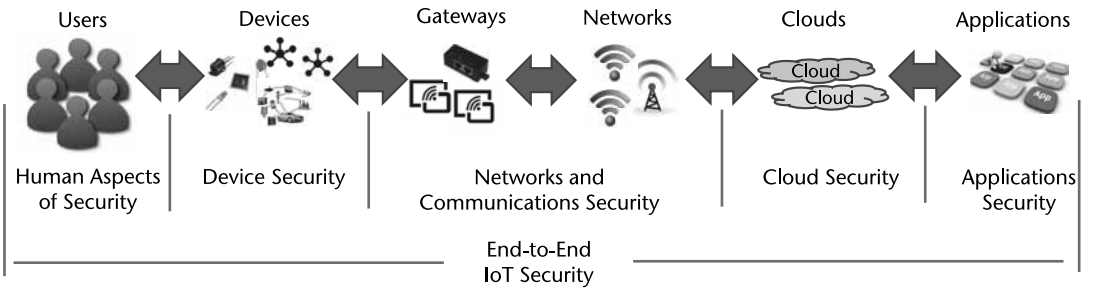

Figure 7.1 End-to-end IoT security requires security mechanism across multiple assets and IoT components.

7.1.2 Security Drivers for Modern IoT Systems

Modern IoT systems are characterized by increased sophistication and intelligence. The latter stems from the deployment of large numbers of internet-connected devices, including smart objects with semiautonomous behavior such as drones and automated guided vehicles. The rising sophistication of IoT systems and applications introduces new security challenges, yet it also enables innovative functionalities. These challenges include:

- *New ways for launching large-scale security attacks:* The proliferation of IoT devices and their deployment introduces new cybersecurity vulnerabilities and enables new ways of conducting large-scale attacks. As a prominent example, in October 2016, we witnessed the first large-scale distributed denial of service (DDoS) attack based on IoT devices, which took advantage of vulnerabilities (e.g., hard-coded passwords, poorly patched software) of internet-connected closed circuit television (CCTV) cameras and digital video recorders (DVRs). This notorious attack was based on the deployment of the Mirai malware on these devices [1].
- *Vulnerabilities in IoT devices and smart objects:* As IoT devices become more sophisticated and interconnected, hackers are provided with new opportunities (e.g., backdoors) to attack them. At the same time, the cyber resilience of these devices becomes more significant for the applications that they support, when compared to the resilience of simpler devices (e.g., passive sensors). Nowadays, adversaries attempt to exploit vulnerabilities associated with complex devices, which can cause significant damage. For instance, in July 2015, 1.4 million cars that were recalled by a major manufacturer due to potential hacking of their control software.
- *Threats associated with operational technology (OT) and physical security:* Industrial IoT (IIoT) systems bring together information technology (IT) and OT. This asks for new integrated approaches to security that protect both cyber and physical assets. OT systems are characterized by poor cybersecurity, as OT protocols are usually vendor-specific and not designed for security. Moreover, OT environments are sometimes supported by old and insecure IT systems (e.g., old operating systems, vulnerable drivers' software) and come with very poor documentation regarding their security features. Hence, malicious parties try to compromise physical assets (e.g., data centers) in order to attack cyber assets. In some cases, they also attempt combined attacks, which go against cyber and physical assets at the same time.
- *Attacks against digitally interconnected IoT infrastructures:* IoT systems are gradually becoming digitally interconnected, as a part of value chains in different sectors. For example, IIoT assets in factories and logistics are getting interconnected as part of the manufacturing value chain. Similarly, IoT-enabled healthcare platforms connect to IoT-based intelligent transport platforms to enable the delivery of emergency services. The interconnection of different IoT infrastructures increases their dependencies and provides the means for attacking one infrastructure by launching an attack to an

interconnected infrastructure. Once one infrastructure is attacked, its interconnected counterparts suffer from cascading effects.

- *Wide spectrum of IoT-related attacks:* Nontrivial IoT systems comprise many IT elements, such as networking devices, computing systems, middleware systems, cloud computing systems, and many different internet-connected devices. Therefore, IoT security solutions must protect a wide range of assets against many different attacks. Typical examples include scanning attacks against wireless networks, protocols' attacks, data theft, loss of confidentiality attacks, attacks against cryptographic algorithms, spoofing attacks, attacks against operating systems and applications, denial of service (DoS) attacks, jamming attacks, access control attacks, and physical security attacks. As a prominent example, Figure 7.2 illustrates the creation of a botnet, which is nowadays the prevailing mechanism for the facilitation of DDoS. It especially presents how botnets can compromise vulnerable IoT systems and accordingly exploit them to launch DDoS attacks against a target.
- *Volatile and demanding regulatory requirements:* Integrators of IoT solutions and providers of IoT services must nowadays comply with stringent directives and regulations, which they cannot afford to ignore. As a prominent example, in Europe, the General Data Protection Regulation (GDPR) imposes strict requirements for personal data protection, along with extremely high penalties for cases of noncompliance. GDPR has a global impact, as several countries are already using GDPR as a blueprint for updating their data protection laws and regulations. Since 2020, Californians have been protected by the California Consumer Privacy Act (CCPA) [2], a state statute

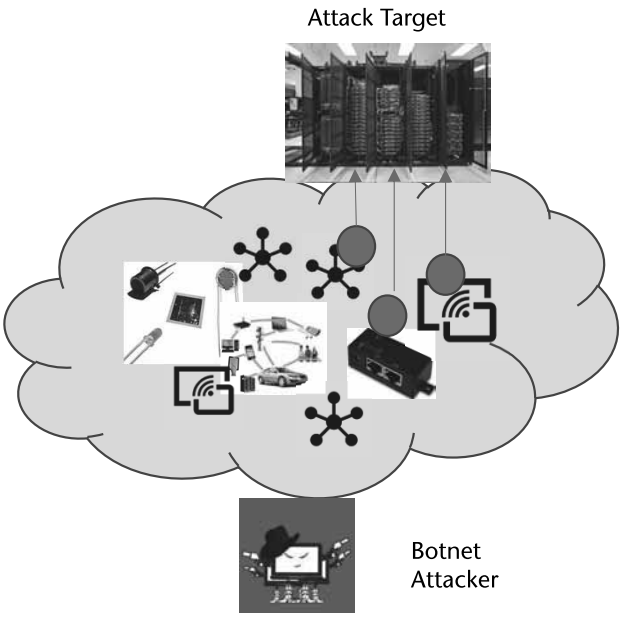

Figure 7.2 The establishment of a botnets is one of the most popular mechanisms for launching DDoS attacks.

intended to enhance privacy rights and consumer protection for residents of California.

7.1.3 Notorious IoT Security Attacks

The presented IoT security challenges have been reflected in various real-life security incidents, including some of the most notorious IoT attacks that have taken place during recent years. Specifically:

- In November 2013, the Linux Darlloz (i.e., a worm for Linux devices) was identified. It affected IoT applications, as several IoT devices are running variations or cut-down versions of Linux.
- In January 2015, the Lizard Stressor attack tools place, when in-home IoT routers and commercial routers were converted to zombies.
- In 2015, a prominent car manufacturer recalled 1.4 million cars that could have been hacked in their control software. This attack alerted the community about the potential severity and complexity of IoT security in the light of emerging applications such as autonomous and connected driving.
- In October 2016, major internet sites (e.g., Twitter, Amazon, Netflix, and Spotify) become unreachable for a considerable amount of time. The source of the problem was a DoS attack, which was launched by adversaries that gained access to numerous IoT devices. This was one of first large-scale IoT DoS attacks on the internet.

These incidents demonstrate that IoT security challenges are for real. They also justify enterprises investments in IoT security and research towards identifying novel and effective security solutions.

7.1.4 Technical Challenges of IoT Security

7.1.4.1 The Unique Nature of IoT Devices

To understand the unique characteristics of IoT security when compared to conventional networking security, it is important to underline the different nature of devices when compared to other computing devices, which are, in most cases, directly addressable (e.g., computers have an internet address). In the case of IoT devices, the device (e.g., sensor, RFID tags, industrial automation device) is not always addressable. Rather, such devices are practically accessible through a gateway (e.g., RFID reader, WSN gateway) or edge server that interfaces and/or controls the devices. Hence, conventional attacks over internet hosts cannot be replicated on some IoT devices.

This difference between conventional networked hosts and IoT devices makes IoT security unique in terms of the following aspects:

- IoT devices are not (always) reachable, as most of the time a device is not connected to the internet.

- IoT services can be lost and stolen, which makes security difficult when the device is not connected to the internet.
- IoT devices lack the computing power needed to execute sophisticated cryptography mechanisms. In general, it is difficult to achieve strong security without adequate processing power.
- IoT devices have a finite life, which means that credentials need to be tied to lifetime.
- IoT devices can be transportable (portable), which makes it easy for them to cross borders and fall in the jurisdiction of different authorities. The latter can possibly apply different laws and regulations.
- IoT devices can be recognized by many readers, which means that different data and services of a device can become accessible to different hosts and gateways.
- Many IoT devices are resource constrained devices (e.g., their CPU capacity is limited). Hence, they also cannot execute sophisticated antivirus software. As a result, IoT devices such as intelligent smart meters and light bulbs are much less protected when compared to state-of-the-art desktop and laptop computers.
- In IoT application scenarios, both encryption key management and identity management are needed. Moreover, these mechanisms must be extremely scalable in order to cope with the multitude of devices that comprise an IoT system.
- Some IoT devices serve as security endpoints, which makes it challenging to create or release protection software for each device. Antivirus software vendors do not have adequate economic incentives to properly protect IoT devices that are not sold or deployed in very large quantities. Hence, for some IoT devices, antivirus and protection capabilities will not become widely available soon.

7.1.4.2 The Implications on IoT Security Tasks

Overall, the peculiarities of IoT devices, along with their large number, make the task of developing IoT security mechanisms more difficult than conventional IT security. Specifically, the following security tasks are complicated:

- *Security analysis:* Security analysis for IoT applications requires collection and processing of data from numerous devices. These data need to be processed along with data from a large number of existing security logs. Hence, IoT security analysis is a very challenging big data problem, which involves much more complex data analytics than in the case of conventional network and computing devices.
- *Vulnerability management:* The task of vulnerability assessment for nontrivial IoT applications must deal with the multitude and diversity of vulnerabilities of IoT devices and services. For example, attackers are offered with

opportunities for attacking device hardware and IoT software components, in addition to conventional cloud and networking components. Hence, processes like pen-testing (i.e., "ethical hacking") towards identifying possible vulnerabilities in an IoT system become more complex and time-consuming.

- *Design and implementation of security controls:* The design and implementation of IoT security controls must cope with a more complex multistakeholder environment. The latter must consider the roles and offerings of new actors such as device vendors and original equipment manufacturers (OEM), in addition to traditional actors such as network operators and cloud services providers. Likewise, the deployment of security controls comprises multiple implementation layers, which increases the complexity of security controls.
- *Security knowledge management:* To support security tasks such as security analysis and implementation of configurable security controls, there is a need to model and manage knowledge about IoT assets such as IoT devices and services. This makes the task of structuring and managing IoT security knowledge very challenging. Next, we discuss the need for extending state-of-the-art security knowledge bases (SKBs) with IoT security information.
- *Scalable security infrastructure:* IoT security asks for IT infrastructures with higher capacity and increased scalability. For instance, security data analytics infrastructures demand more bandwidth and storage, in order to collect and analyze larger amounts of security data.

7.1.4.3 The Cyber-Physical Nature of IoT Systems

One of the fundamental differences of IoT security from other types of cybersecurity systems stems from the cyber-physical nature of IoT systems. Most nontrivial IoT systems are CPS, which interact with the physical world, while providing digital services as well. Hence, they realize a convergence between IT and OT. However, IT and OT differ in terms of their security requirements:

- *IT security (i.e., cybersecurity):* This deals with any-to-any connectivity between digital systems, to ensure their integrity, availability, and confidentiality. There are already many known solutions for cybersecurity and data protection. A typical approach involves putting the system in quarantine until the problem is resolved or even shutting down the system in order to mitigate the cybersecurity problem.
- *OT security (i.e., security of physical systems):* This deals with hierarchical connectivity between field devices, aiming at ensuring their availability, integrity, and confidentiality. Relevant security solutions entail physical access control to the devices, as well as techniques that ensure safety in the field. In several cases, physical systems are characterized from nonstop operations, especially in the case of mission-critical systems. Mission-critical systems should never stop, even in cases where they are breached. For example, this is the case with IIoT systems in energy plants, manufacturing shopfloors, and oil refineries.

Due to their dual nature, IoT systems need to address and combine both physical security and cybersecurity as part of integrated security mechanisms. This is the reason why large-scale CPS (e.g., ports, cities, and factories) must deploy sophisticated security mechanisms beyond conventional cybersecurity.

7.1.4.4 Privacy Challenges

IoT can also raise privacy issues, especially in the case of applications that collect and process personal data or other types of sensitive information. A wide range of IoT devices are used to collect information from a user's personal context. Examples include TV sets, webcams, home thermostats, remote power outlets, door locks, home alarms, and garage door openers. All these devices provide the means for acquiring information about a user's behavior and personal context, which if abused can lead to privacy violations. Personal data that flow across different systems can be compromised. Typical examples of personal data include an individual's name, address, date of birth, health-related data, or even credit card numbers.

In IoT environments, privacy issues can arise as a result of various weaknesses, including:

- Weak authentication mechanisms (e.g., simple passwords allowed);
- Unencrypted communications, which can enable adversaries to eavesdrop personal information;
- Poor user interface security, which can reveal personal data (e.g., personal identification numbers);
- Software updates without encryption that can be intercepted and compromised.

7.1.4.5 Human and Social Aspects of IoT Security

IoT security is not only about technology measures. Rather, the ever-important social organizations and human factors should be considered as well. Successful IoT security solutions must be intuitively useable and acceptable by their users. Furthermore, it is very important to persuade end users (e.g., employees of an organization) to use them. In this context, it is also important that they fulfill the subtle needs of diverse user groups.

In this direction, the development of IoT security solutions should address the nontechnical aspects such as:

- The usability of the security solutions, including, for example, the learnability, efficiency, error tolerance, and memorability of the solutions;
- The context of their use, such as different work settings, task goals, organizational processes, and the wider physical and social environment where they are deployed;
- Various user experience factors, including, for instance, trust and accountability.

Emerging IoT systems must be also trusted and acceptable by their users. To this end, IoT security integrators are putting emphasis on the integration of user control and transparency mechanisms in their IoT system. Such mechanisms are important towards gaining the confidence of everyday users and citizens in IoT solutions.

Overall, beyond technology considerations, the development of IoT security solutions should also deal with the acceptance of the solutions by their end users. Hence, User-Experience (UX) and Human Computer Interaction (HCI) activities must be undertaken as part of the development of effective IoT security solutions. The need for such activities increases the complexity and sophistication of IoT system, as the interaction of users with devices must be considered as well. In recent years, various research projects (e.g., FP7 SOCIETAL [3], H2020 GHOST [4]) have demonstrated IoT security solutions that boost users' confidence and feature exceptional usability. In the near future, such solutions will move from the realm of research labs to the enterprise.

7.2 IoT Security Risk Assessment

7.2.1 Overview of the Risk Assessment Process

One of the most important security management activities for IT and IoT systems is the assessment of their security risks. Security risk assessment helps organizations identify and mitigate the operational and the economic impact of security threats. Security risk assessment processes identify and assess the probabilities of security vulnerabilities and threats, as a means of anticipating them and taking proper measures to prevent them. When carrying out a security risk assessment, organizations examine their assets end to end, while considering the attacker's perspective. The outcome of the risk assessment process is typically a comprehensive assessment report that supports security experts and the business management of an organization in their decision making. Specifically, the report lists the risks that are associated with the various assets of the organization and outlines the likelihood of their occurrence. Based on this information, managers can make informed decisions about resource allocation, procurements of security solutions, and how to select and best implement security controls.

A typical security risk assessment process entails the following steps:

- *Assets identification:* This is identifying the assets that are most valuable to the company. In the case of IoT security risk assessment, the assets include IoT devices and IoT services.
- *Threats identification:* This is determining the potential threats to each asset. This step can take into account information about each IoT asset, such as information known from an SKB or information collected from a device via an appropriate probe.
- *Likelihood estimation:* This is estimating the likelihood that events associated with the threat will occur. In most cases, a probability estimate is

produced based on some deterministic or stochastic model about the asset and the attacker's behavior.

- *Impact assessment:* This is determining the impact criticality of each event. Risks that have a very high potential impact on the organization should be addressed as a matter of priority.
- *Risk scoring:* This is calculating risk scores based on combinations of information about events' probabilities and their impact. The worst risks are obviously the ones that have a high probability of materializing, while at the same time having a high impact on the organization.

The risk assessment process may also include activities such as identification of threats, profiling of assets, and definition of risk mitigation actions. The latter are very important in order to alleviate the potential risks. Overall, the risk assessment process is driven by specific business and security objectives, which prioritize the assets and threats that are most critical from the business perspective. Figure 7.3 illustrates the main steps of a typical risk assessment process. The process is driven by business drivers and objectives. It includes asset identification and profiling, threat identification, risk assessment, and risk mitigation steps. These steps are part of most risk assessment methodologies.

Most security risk management methods provide tools and techniques for the estimation of risk situations for various assets. These techniques combine information about the processes, the assets, and the infrastructure of the organization towards producing an estimate of the likelihood and importance of the risk. Once potential risks are identified and graded, appropriate protective measures can be designed. In practice, a risk assessment report includes a list of risks or threats to a system, together with their probabilities estimates.

Figure 7.3 The main steps of a typical risk assessment process.

7.2.2 Risk Management Standards

IoT security risk assessment can be based on standards-based frameworks for risk management. Over the years, various standardization organizations have developed risk management standards that support the identification of risks and the assessment of their probabilities. There are standards that provide general considerations and guidelines for risk management processes. For example, the ISO 31000 standard [5] provides general principles and guidelines for effective risk management. It also outlines a generic approach to risk management, which is applicable to different types of risks (e.g., financial, safety, project risks) and to different types of organizations. Likewise, the IEC 31010 standard [6] provides guidance on the selection and application of systematic techniques for risk assessment. The guidance is provided in the general context of the ISO 31000 standard.

Apart from these general-purpose risk management standards, there are standards that include specific guidelines for the IT sector. Prominent examples include:

- *ISO 20000 [7]:* This describes best practices for IT Service Management (ITSM). It supports organizations in delivering managed services, measures service levels, and assesses their performance. The standard is strongly linked to ITIL (i.e., the IT Infrastructure Library framework), which is one of the most prominent state-of-the-art approaches for IT service management worldwide. The standard is not focused on risk assessment, yet it sets the context for addressing the risks of IT service delivery.
- *ISO 27000 [8]:* This is a series of standards that are designed to assist companies in managing cyberattack risks and internal data security threats. The most popular standard of the family is ISO 27001, which provides specifications for an information security management system (ISMS). The latter includes policies and procedures for information risk management based on legal, physical, and technical controls. It is a very popular standard that is widely used for auditing and certifying the IT security processes of organizations. Within the ISO 27000 family of standards, ISO/IEC 27002 [9] outlines best practices for implementing information security controls and ISO27005 [10] describes how to conduct an information security risk assessment. Both ISO/IEC 27002 and ISO 27005 have been developed in accordance with the requirements of ISO 27001.
- *The common methodology for IT security evaluation [11]:* This ensures that the process of specification, implementation, and evaluation of a computer security product has been conducted in a rigorous, standard, and repeatable manner. Moreover, the methodology ensures that the implementation of the security product is aligned to the requirements of its target environment and use.

These standards are applicable to all organizations that provide or use IT products and services. Most of them specify framework conditions for the risk management process, which can be applied in IoT environments as well. However, they do not provide in-depth information and details on how to implement a risk

assessment process, such as information about how the assets will be modeled and how the risk probabilities will be assessed. Hence, when following the above-listed approaches, there is a need for detailing more specific methods for risk analysis, assessment, and evaluation. This is the reason why there is a rich research literature about concepts, algorithms, and tools for protecting IT systems and assessing their risks.

Most of these methods introduce techniques for quantitative risk assessments, including methods that consider monetary costs and financial damage [12, 13]. For example, this is the case with the French EBIOS (Expression des Besoins et Identification des Objectifs de Sécurité - Expression of Needs and Identification of Security Objectives) and the United Kingdom's CRAMM (CCTA Risk Analysis and Management Method) methods, as well as with recent updates in the ISO/IEC 27005 standard. Most of the risk assessment methods and tools calculate risk estimates based on the simple formula [14, 15]:

$$\text{risk} = \text{probability} \times \text{potential damage}$$

The terms and scales for the assessment of the risk probabilities and of the potential damage are predefined and vary for the different methods. For example, different scales are applied in the NIST Risk Management Guide [16] and in the Mehari method [17]. The selection of a specific risk assessment tool is based on practical considerations and depends on how well the terminology of the application can match the predefined terminology of the selected risk assessment methodology.

There are also tools and techniques for general-purpose risk assessments, including low-level mechanism and tools, beyond the high-level guidelines of standards-based frameworks. Some prominent examples include:

- *The AURUM system [18]:* This provides a graphical tool for the risk modeling based on ontologies. It uses a Bayesian approach for determining threat probabilities [19].
- *The OCTAVE method [20]:* This is based on subjectively estimated probabilities. It can therefore be considered as an a priori distribution with regards to the Bayesian approach.
- *The CORAS method [21]:* This facilitates the integration of different risk assessment processes. In each of the methods, the identification of the probability of an attack is not done automatically, but a priori to any risk assessment.

Furthermore, a variety of mathematical models that embed different reasoning techniques have been proposed such as integer linear programming, constraint programming, stochastic optimization, multicriteria optimization, robust optimization, and metaheuristics. Several optimization and decision support approaches are used for risk assessment. They integrate techniques and models from AI, operations research, and computational logics (e.g., [22–24]).

7.2.3 Peculiarities of IoT Security Risk Assessment

The previously outlined risk assessment methods are deployed and used by various organizations. They fulfill various cybersecurity risk assessment needs. In most cases, these methods can be customized and used for IoT security risk assessment as well. Nevertheless, this customization should consider the peculiarity of IoT systems, including the special nature of IoT devices and the cyber-physical properties of IoT environments. In this direction, the following guidelines should be considered:

- *Perform continuous and frequent risk assessments:* This is needed in order to cope with the dynamism, variability and scale of IoT deployments. The latter comprise very large numbers of devices. In several cases, these devices are also densely interconnected. Hence, the outcomes of a risk assessment process are bound to change frequent. By performing continuous and frequent risk assessments, organizations can remain up to date about the risk status of their IoT assets.
- *Collaborative risk assessment across stakeholders of IoT value chains:* IoT-based value chains (e.g., virtual manufacturing chains) interconnect various IoT systems. In such cases, the risk status of an asset of any given organization depends on the status of other assets that belong to interconnected organizations. For instance, in a smart manufacturing scenario, the risks of the IoT services deployed by manufacturer depend on the risk status of the IoT assets of its supplier. In this context, a collaborative risk assessment process that involves both the manufacturer and its supplier can lead to increased preparedness and accuracy. Such collaborative functionalities are hardly provided in the scope of traditional risk assessments.
- *Consideration of IT-OT integration, including the interrelationships between communication, computing, and control technology:* This is a good practice that aims at addressing the cyber-physical nature of IoT systems. The latter comprise cyber and physical assets, which must be both considered in the scope of the risk assessment process.
- *Enhanced knowledge modeling, as part of an SKB:* SKBs play an important role in the security risk assessment process. They provide the means for matching events against known attack patterns. This matching requires for example well-structured knowledge about the vulnerabilities of IoT devices, as well as about how threats and vulnerabilities on one IoT asset affect another. Conventional cybersecurity knowledge bases do not comprise adequate information about IoT assets. Hence, their use in IoT security risk assessment can lead to mistakes such as missing some important risks. Thus, there is a need for structuring and using enhanced SKBs that include information about IoT assets and vulnerabilities.
- *Automation of security risk assessments:* IoT security risk assessment can benefit from automation, towards coping with the complexity and scale of IoT systems. Automation can be based on software systems such as machine learning and AI. It can process large numbers of security events in short

timescales, which is much more effective than conventional manual assessments of IT assets.

- *Adaptation to the IoT environment:* State-of-the-art risk assessment frameworks need to be customized to the peculiarities of IoT organizational environments. For example, this implies a need for considering the role of IoT stakeholders like device integrators and OEMs.

7.3 IoT Security Solutions

7.3.1 IoT Security Architectures

The importance of IoT security has given rise to the inclusion of security capabilities and functionalities in most popular reference architectures and frameworks for IoT systems, including the architectures that we presented in Chapter 1.

7.3.1.1 The Industrial Internet Security Framework (IISF) of the Industrial Internet Reference Architecture (IIRA)

As outlined in Chapter 1, the IIRA specifies a common architecture framework for developing interoperable IoT systems for different vertical industries. Based on the analysis of multiple use cases in different sector, the IIRA presents the structure of IoT systems from four viewpoints, namely, business, usage, functional, and implementation viewpoints.

The IISF [25] provides the security view of the IIRA viewpoints (i.e., it is a security framework for systems built based on the IIRA). The IISF secures all the functional building blocks of the functional viewpoint of the IIRA end to end, including field, edge, and cloud endpoints. One of the key functionalities of the IISF is security monitoring and analysis, which is destined to:

- Capture data about the overall state of the system from various endpoints and connectivity traffic;
- Analyze it to detect possible security violations or system threats;
- Initiate (upon the detection of a violation or threat) one or more actions for protection in line with the system's security policy.

Overall, the IISF provides a blueprint for developing, deploying, and operating IoT security systems, including systems that protect all the possible endpoints, as well as the networking and communication infrastructure of IoT systems. Moreover, the IISF specifies functionalities for security analytics, as well as for the configuration and enforcement of IoT security policies.

7.3.1.2 Security Functionalities in the OpenFog Reference Architecture

The OpenFog Consortium was a consortium of high-tech industrial enterprise companies and research or academic institutions, which were collaborating towards

standardizing and promoting the fog computing paradigm.[1] Fog computing is directly associated with IoT as introduced in Chapter 4. The reference architecture of the OpenFog Consortium illustrates the structure of fog computing systems. It presents how fog nodes can be connected partially or fully in order to enhance the intelligence and operation of an IoT system. Moreover, it presents solutions about growing system-wide intelligence away from low-level processing of raw data. The reference architecture is described in terms of different views, including functional and deployment views. The reference architecture specifies a range of cross-cutting functionalities as well (i.e., functionalities that are applied across all layers of an OpenFog-compliant system). These cross-cutting functionalities are conveniently called perspectives.

Security is one of the perspectives of the OpenFog reference architecture. This perspective outlines the importance of end-to-end security as a cross-cutting function of IoT systems. Security is integral to all IoT scenarios and is defined end to end, that is, it covers the underlying silicon (fog or device) and all upper software layers of the fog architecture. This is because if one of the layers is not secure, the entire solution is not secure. Hence, end-to-end security covers everything between the cloud and the things on the edge of the network. According to the OpenFog reference architecture, security starts with the individual fog node hardware. Once the trustworthiness of fog nodes has been ensured, other secure layers (e.g., a secure fog network) can be deployed on top of the nodes as a means of end-to-end security that provides secure node-to-node, node-to-thing, and node-to-cloud communication.

7.3.1.3 ISO/IEC CD 30141 Internet of Things Reference Architecture (IoT RA)

ISO/IEC CD 30141 Internet of Things Reference Architecture (IoT RA) specifies another standards-based reference architecture. Its specification was based on a heuristic analysis of system characteristics that are common in most IoT systems (e.g., auto-configuration, discoverability, scalability). From them, it proposes a conceptual model where typical IoT entities or actors are their relationships are described. The reference model considers security aspects both as part of the inside-domain and cross-domain functions. Security implementation in the scope of this architecture is based on the following principles:

- Safety and security at sensing and control domain must be enforced by edge entities.
- Security, safety, resilience, trust, and privacy are functions present in all the domains (cross-domain). Their role is to ensure data privacy protection, confidentiality, integrity, authenticity and confirmation of exchanged information, fault tolerance, and self-healing.

1. In 2019, the OpenFog members became members of the Industrial Internet Consortium and the assets and obligations of OpenFog were transferred to the Industrial Internet Consortium: https://www.iiconsortium.org/IIC-OF-faq.htm.

- Authentication mechanisms, which consider multiple levels of trust, are the proposed way of protecting data confidentiality.

7.3.2 Firmware Over The Air (FOTA) Mechanisms

FOTA mechanisms provide a means for addressing the security peculiarities of IoT devices. They enable the implementation of IoT security mechanisms at the firmware level. FOTA mechanisms include updates to the firmware of mobile phones (e.g., towards ensuring software bug fixes). In this case, delta encoding technologies (i.e., compression) are also used towards reducing the size of the patch and keep the overall FOTA mechanism efficient. This approach is quite similar to the way that several web browsers are updated. FOTA mechanisms can be applied to IoT devices directly or through the IoT gateway to which the devices are attached.

7.3.3 Network-Level IoT Security

Network-level security is a critical component of IoT security mechanisms. Strong network security is very important given that it limits the spectrum of options available to attackers. By defending the network layer of an IoT deployment, it is possible to capture attacks before affecting other elements of an IoT system.

Some of the most prominent network-level security measures for IoT systems include:

- *Network monitoring and analysis of network traffic using popular tools:* For example, tools such as NetFlow [26] are used to analyze network traffic flow and volume and to determine where traffic originates from, to where it is destined, and how much traffic is being generated.
- *Detection of anomalies and behavior patterns that could indicate network-level security attacks:* Such patterns can be identified based on mining network traffic data. Moreover, they can be matched to known vulnerabilities and attacks using an SKB.
- *Analysis of security events in terms of their root causes, timing, and stakeholders:* This is also part of security analytics for IoT, which can take advantage of data-mining mechanisms, including machine learning.
- *Monitoring and extraction of statistics (e.g., number and types of devices, percentage of nonencrypted traffic, lists of the most active devices) to benchmark the normal operation of the network:* Following this benchmarking, it is possible to identify significant deviations from the normal operation or other patterns that may reveal problems and abnormalities.

7.3.4 Multilevel Security for IoT and CPS

Beyond the network level, security should be applied at all other levels of an IoT system. In several cases, security mechanisms should consider the cyber-physical

nature of IoT systems as well. Hence, security mechanisms are typically applied across four different layers including:

- *The enterprise network layer:* Here IT applications such as Enterprise Resource Planning (ERP) are executed. This layer is nowadays implemented in the cloud and hence cloud security mechanisms need to be applied.
- *A demilitarized zone (DMZ) layer:* This aims at protecting information, digital assets, and access to systems flowing or residing in between the digital (IT) and physical (OT) worlds.
- *The supervisory layer:* This implements the access to the automation and control layer and makes provisions for the synchronization between the physical and the digital worlds.
- *An automation and control layer:* This deals with field level functionalities.

Different security mechanisms apply to each one of the physical and digital worlds (i.e., to OT and IT). Cloud security, network security, and application control are applicable to the cyber layer only, while configuration management for security applies to the physical world. Moreover, there are secure access and identification services that are implemented and deployed across both the digital and physical worlds.

7.3.5 IoT Security Systems Monitoring and Threat Intelligence

There are also security solutions for monitoring IoT systems towards obtaining and analyzing security information about them. For example:

- Fireeye's OpenIOC suite of tools enables the sharing threat intelligence information. It comprises an extensible XML schema that enables one to describe the technical characteristics that identify a known threat, an attacker's methodology, or other evidence of compromise.
- Structured Threat Information Expression (STIX) is a language and serialization format used to exchange cyber-threat intelligence (CTI). STIX enables organizations to share CTI with one another in a consistent and machine-readable manner. Hence, it allows security communities to understand better what computer-based attacks they are most likely to see and to anticipate or respond to those attacks faster and more effectively. Based on STIX, other formats for cyber-threat intelligence has been derived, such as the FINSTIX format for sharing security information in the financial sector [27, 28].
- IOTDEF (IoT Defense) is an IoT defense framework, which is simple, affordable, and delivered as a service (i.e., it is a Security-as-a-Service (SECaaS) framework). It is in practice an auto-configurable smart firewall for IoT devices. It has been instantiated for different applications as in the case of the MyRattrap framework for security in smart home environments.

7.3.6 OWASP IoT Privacy Recommendations

To alleviate privacy challenges and concerns, the Open Web Application Security Project (OWASP) has provided a number of recommendations that cover different areas and concerns. In particular:

- *Authentication:* OWASP recommends the use of strong passwords, the implementation of granular access control in line with the confidentiality requirements of operations and datasets, as well as the establishment of mechanisms for protecting authentication credentials. Furthermore, it recommends a two-factor authentication where practical, along with secure password recovery mechanisms. It is also advised that application designers and developers make provisions for reauthentication in cases of sensitive features. Also, password control configuration options should be provided.
- *Transport encryption:* OWASP recommends encrypting data when transiting networks, including the use of Secure Sockets Layer (SSL) and Transport Layer Security (TLS) or other industry standards if these are not available. It also recommends avoiding the use of proprietary encryption mechanisms.

OWASP provides privacy-related recommendations associated with the web user interface and the software and firmware updates of IoT systems. In particular:

- *Concerning the web user interface:* OWASP recommends changing default passwords during the initial setup, as well as default usernames. It also highlights the importance of implementing robust password recovery mechanisms, along with mechanisms for protecting from Cross-Site Scripting (XSS), SQL Injection (SQLI), and Cross-Site Request Forgery (CSRF). It is also advised not to expose credentials in network traffic and to require strong passwords. Moreover, accounts should be automatically locked after three to five failed logins.
- *Concerning software and firmware updates:* OWASP recommends ensuring that updates are possible, but also are able to encrypt the update files. Moreover, it is advised that transfers are updated over encrypted connection and to ensure that update files do not expose sensitive information. Furthermore, updated files should be verified before uploading and applying, while the update server should be secured.

7.3.7 SKBs

As already discussed, SKBs are very important and versatile tools for matching identified threats and incidents to existing knowledge SKB consolidate several sources of knowledge for CTI, such as:

- *Common Platform Enumeration (CPE):* This is a structured naming scheme for IT software, systems, and packages.

- *Common Weakness Enumeration (CWE):* This lists common software's vulnerabilities.
- *Common Attack Pattern Enumeration and Classification (CAPEC):* This lists common attack patterns on software and their taxonomy.
- *Common Vulnerabilities and Exposures (CVE):* This lists all publicly known cybersecurity vulnerabilities and exposures.

SKBs can also collect and store external CTI data sources through available documents in various formats such as JavaScript Object Notation (JSON) and eXtensible Markup Language (XML). There are several SKBs, including commercial SKBs from major security vendors and SKBs from standards development bodies (e.g., the OWASP Security Knowledge Framework). Nevertheless, these SKBs are general purpose and do not comprise suitable security knowledge for IoT assets, which is a setback to supporting applications such as risk analysis and security management activities for IoT environments.

To enable effective and automated risk assessments for IoT systems, SKBs should also collect and consolidate information from additional sources, as means of covering IoT assets. For example, the Open Vulnerability and Assessment Language (OVAL) specifications enable the description of IoT assets.

7.3.8 Supply Chain Security

To provide security solutions across interconnected IoT systems, security reference models such as the IISF provide insights on how to implement general-purpose security measures for large-scale IoT systems. Furthermore, they provide ideas and guidelines about implementing security measures across entire supply chains that span multiple IoT platforms. In an era where infrastructures are becoming digitally interconnected, an attack on one part of the supply chain can have severe effects on others. The issue of supply chain security is known, and relevant standards exist. For example, the ISO 28000 [29] standard specifies the requirements for a security management system, including those aspects critical to security assurance of the supply chain. Hence, to address supply chain security, there is a need for seamless information sharing and collaboration between supply chain stakeholders and their IoT systems.

7.3.9 Decentralized Secure Data Sharing for Security in IoT Value Chains

To secure IoT-based supply chains, it is essential to facilitate information sharing between supply chain participants. The most prominent way for sharing information between the various stakeholders involves the use of a shared database, where selected information from all the different supply chain participants will be persisted and accessed. Nevertheless, centralized information sharing is associated with prominent limitations including:

- *The need for a trusted third party (TTP) that owns and operates the centralized database infrastructure:* However, in IoT supply chains, there is not always a prominent and accepted TTP.
- *Persisting shared information in a centralized infrastructure makes it susceptible to cybersecurity attacks:* Indeed, a centralized database, once compromised, would infect and destabilize the entire information-sharing system.

In recent years, decentralized paradigms to sharing information have emerged, including blockchain-based approaches. Blockchains present some unique opportunities for securely sharing information across parties such as [30]:

- *A distributed architecture that can deter or minimize the effect of cyberattacks against the shared information:* A distributed network structure provides inherent operational resilience as there is no single point of failure.
- *A consensus validation mechanism for news blocks of (shared) data:* This enables a continuous check on the integrity of past transactions and guarantees strong security and decentralized trust. In the scope of a blockchain mechanism, there is no need for a TTP to authenticate and validate ownership of the data, as the nodes in the overall network use the peer-to-peer network to process and verify transactions.
- *Strong encryption features:* As blockchain networks employ multiple forms of encryption at different points and provide multilayered protections against cybersecurity threats. Participant access rights are secured through asymmetric-key cryptography or public and private key encryption.
- *Transparency, which provides another degree of cybersecurity protection:* It is more challenging for hackers to place malware in the network to collect information and to transmit it to another database managed by the hacker [31]. Each participant has an identical copy of the ledger, and hence the network creates the opportunity for deploying enhanced compliance processes including, among others, real-time auditing or monitoring. As a result, vulnerabilities and threats may be identified quickly if good risk management and compliance controls are implemented.
- *Opportunities for securing connected devices and sharing information from them:* This is done through making them peer nodes in the blockchain network [32].
- *Support for smart contracts:* This can implement security logic (i.e., security workflows) over the shared information.

In practice, these benefits can be better leveraged in the scope of permissioned blockchain networks [27, 33] rather than in the scope of the public blockchain infrastructures that support blockbuster cryptocurrencies such as Bitcoin and Ethereum. The main reason for this is that permissioned blockchains offer authentication and privacy control of the participants, along with much better performance (i.e., reduced latency and handling of more transactions per second). The latter benefits stem from the fact that permissioned blockchains are decoupled from the

computationally expensive proof-of-work mechanisms. Moreover, permissioned blockchains are often hosted on cloud platforms that have robust cybersecurity controls across different layers of the technology stack.

The use of permissioned blockchain networks for sharing security information is not widespread and security implementations are in early stages. For example, there are early implementations of blockchain-based solution for sharing security information across financial institutions [27]. There are also blockchain solutions for information sharing in general, as well as for protecting specific IT systems and IoT devices (e.g., [32]) and enabling the trusted exchange of data between them (e.g., [34]).

Security is not the only area where blockchain infrastructures could add value to IoT systems. As discussed in Chapter 9, the implementation of decentralized IoT applications over blockchain networks can offer scalability benefits as well. In the future, blockchain infrastructures could offer a compelling alternative to the mainstream edge or cloud architectural paradigm for IoT applications.

7.3.10 Machine Learning and AI Solutions for Data-Driven IoT Security

Machine learning and AI are already used to analyze large amounts of security data, towards deriving insights on potential security vulnerabilities and risks. IoT devices and infrastructures generate large volumes of data and hence can be analyzed based on similar approaches. Such analysis can be based on the execution of machine learning algorithms over IoT security datasets. In this context, the use of deep learning for detecting anomalies and potential intrusions is suggested given that the performance of deep learning algorithms improves significantly when trained with very large datasets. The research literature includes many works that apply machine learning and deep learning techniques towards detecting security issues and anomalies at the network level (i.e., techniques applied over network layer datasets). Nevertheless, such data mining techniques have not been extensively applied in the case of IoT applications. The growing popularity of machine learning, deep learning, and AI provides opportunities for building data-driven systems that will detect security issues timely and effectively [35, 36].

7.3.11 Identify and Access Management (IAM)

IoT systems must also implement solutions that identify the users of IoT devices, manage their identities, and control access to the IoT functions and services. Such solutions are commonly called IAM solutions. IAM solutions for IoT systems must deal with the dynamicity and diversity of IoT environments, where different IoT devices may be deployed or undeployed. Therefore, modern IAM solutions for IoT must provide support for:

- Secure addition of new devices at short time scales;
- Granular access to the various functionalities of IoT devices (some users may not be authorized to access the full range of available functionalities);

- Access to and sharing of data from IoT devices within an organization and across different organizations;
- Privacy control features in all cases where sensitive data are collected, stored, or shared with other users and devices.

7.3.12 Data Encryption and Decryption Solutions

Encryption can be considered as a cryptographic service. It masks information in ways that make it impossible for unintended parties to read or interpret it. Hence, encryption services protect the confidentiality of the information from potential eavesdroppers. Likewise, it makes sure that the information can be deciphered by intended parties only. There is a very wide range of encryption algorithms, including symmetric and asymmetric algorithms. In symmetric encryption, both the sender and the receiver use an identical cryptographic key. However, in asymmetric encryption, there are two different keys: one public and the other private. The two keys form a unique pair that can encrypt and decrypt the transmitted information.

In both symmetric and asymmetric encryption, a cryptographic key and the unprotected data are given to an encryption algorithm. The latter ciphers (i.e., encrypts) that data, which ensures its protection from eavesdroppers. Likewise, the receiving party uses a key to decrypt the data that it received.

Recently, there have been significant advancements in privacy-preserving encryption methods, which are gradually moving from research implementations to enterprise deployments. For example, functional encryption solutions enable the implementation of IoT services that cannot read the encrypted data, yet they are able to read the results of the computation of specific functions applied to the data. In this way, functional encryption can automatically safeguard GDPR requirements, such as the need to use private data only for the purpose for which they have been provided. As another example, homomorphic encryption solutions enable a service to perform computations on encrypted private data and then return the encrypted result to the user, who can decrypt it to read the results.

7.4 Summary

IoT security is an integral element of every IoT system. This chapter introduced the security challenges of IoT systems. It illustrated why and how IoT security is different from cybersecurity for conventional networked computers and IT systems. The cyber-physical nature of IoT systems and the unique characteristics of IoT devices introduced challenges that cannot be addressed by legacy cybersecurity mechanisms [28]. Emphasis has been paid in outlining the security risk assessment process, along with standards-based frameworks for security risk assessment. These frameworks fall short when it comes to addressing IoT and CPS. Hence, they must be enhanced in various directions, including the ways that they model security knowledge and the frequency of the assessments.

This chapter also introduced a rich set of IoT security solutions, including solutions for analyzing security information, network-level security solutions, solutions

for securely sharing information across IoT systems, IoT data encryption solutions, and solutions for identity management and access control. Moreover, various IoT privacy solutions and recommendations have been outlined. This chapter also illustrated how some of the prominent reference architectures for IoT systems deal with security aspects and building blocks.

Overall, IoT solution architects and integrators cannot afford to ignore the security aspects of their IoT systems. Similarly, managers should acknowledge the role of IoT security as one of the most critical parts of their IoT deployments. Hence, they should be also enthusiastic about investing in strong IoT security rather than treating it as a defensive investment that they would rather avoid.

The implementation of IoT security systems could greatly benefit from the adoption and implementation of security standards. Earlier, we described some of the main IoT security standards. Chapter 8 is devoted to a comprehensive presentation of IoT standards, including mainstream security standards.

References

[1] Soldatos, J., "Why the DDoS Attack Happened and What Can Be Done to Prevent More Episodes," *The Internet of All Things*, October 2016, https://www.theinternetofallthings.com/why-the-ddos-attack-happened-10262016/.

[2] California Consumer Privacy Act (CCPA) of 2018, https://oag.ca.gov/privacy/ccpa.

[3] Seventh Framework Programme, Sociotal, https://cordis.europa.eu/project/id/609112.

[4] Ghost, https://www.ghost-iot.eu/.

[5] International Standardization Organization, "ISO 31000: Risk Management—Principles and Guidelines," Geneva, Switzerland, 2009.

[6] International Standardization Organization, "ISO 31010: Risk Management—Risk Assessment Techniques," Geneva, Switzerland, 2009.

[7] International Standardization Organization, "ISO 20000: Information Technology Service Management," Geneva, Switzerland, 2005.

[8] International Standardization Organization, "ISO 27001: Information Security Management System Requirements," Geneva, Switzerland, 2013.

[9] International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), "ISO/IEC 27002 Information Technology—Security Techniques—Code of Practice for Information Security Management," 2005.

[10] International Standardization Organization, "ISO 27005: Information Security Risk Management," Geneva, Switzerland, 2011.

[11] Common Criteria Working Group, "Common Methodology for Information Technology Security Evaluation—Evaluation Methodology," CCMB-2007-09-004, 2007, http://www.commoncriteriaportal.org.

[12] Peltier, T. R., *Information Security Risk Analysis*, Boca Raton, FL: Auerbach Publications, 2001.

[13] Schechter, S. E., "Computer Security Strength and Risk: A Quantitative Approach," Ph.D. thesis, Harvard University, May 2004.

[14] European Network and Information Security Agency, "Inventory of Risk Management/Risk Assessment Methods," 2010, rm-inv.enisa.europa.eu/rm_ra_methods.html.

[15] Common Criteria Recognition Arrangement (CCRA) Working Group, "Common Criteria for Information Technology Security Evaluation," 2006, www.commoncriteriaportal.org.

[16] Stoneburner, G., A. Goguen, and A. Feringa, "Special Publication 800-30: Risk Management Guide for Information Technology Systems," National Institute of Standards and Technology, 2002.

[17] Clusif Methods Commission, "MEHARI V3 Risk Analysis Guide," 2004.

[18] Ekelhart, A., S. Fenz, and T. Neubauer, "Automated Risk and Utility Management," *Proc. of 6th International Conference on Information Technology: New Generations*, IEEE Computer Society, 2009, pp. 393–398.

[19] Foroughi, F., "Information Security Risk Assessment by Using Bayesian Learning Technique," *Proc. of the World Congress on Engineering, Bd. 1, International Association of Engineers*, 2008, pp. 2–6.

[20] Alberts, C. J., and A. Dorofee, *Managing Information Security Risks: The Octave Approach*, Reading, MA: Addison-Wesley Longman, 2002.

[21] EU Project Nr. IST-2000-25031, "CORAS—Risk Assessment of Security Critical Systems," 2003, http://www2.nr.no/coras/.

[22] Milano, M., (ed.), *Constraint and Integer Programming: Toward a Unifying Methodology*, Boston, MA: Kluwer Academic Publishers, 2004.

[23] Hooker, J. N., "A Hybrid Method for the Planning and Scheduling," *Constraints*, Vol. 10, No. 4, 2005, pp. 385–401.

[24] Rousseau, L. -M., et al., "Solving VRPTWs with Constraint Programming Based Column Generation," *Annals OR*, Vol. 130, No. 1-4, 2004, pp. 199–216.

[25] Industrial Internet Consortium, *The Industrial Internet Security Framework Technical Report*, 2016, https://www.iiconsortium.org/IISF.htm.

[26] Manage Engine, "NetFlow," 2020, https://www.manageengine.com/products/netflow/index-new.html.

[27] Karagiannis, I., et al., "Blockchain Based Sharing of Security Information for Critical Infrastructures of the Finance Sector," *IOSec/MSTEC/FINSEC@ESORICS*, 2019, pp. 226–241.

[28] Soldatos, J., J. Philpot, and G. Giunta, (eds.), *Cyber-Physical Threat Intelligence for Critical Infrastructures Security: A Guide to Integrated Cyber-Physical Protection of Modern Critical Infrastructures*, Boston, MA: now publishers, 2020, https://www.nowpublishers.com/Article/BookDetails/9781680836868.

[29] International Standardization Organization, "ISO 28000:2007 Specification for Security Management Systems for the Supply Chain," September 2007, https://www.iso.org/standard/44641.html.

[30] Kshetri, N., "Blockchains Roles in Strengthening Cybersecurity and Protecting Privacy," *Telecomm. Policy*, Vol. 41, No. 10, 2017, pp. 1027–1038.

[31] Gu, J., et al., "Consortium Blockchain-Based Malware Detection in Mobile Devices," *IEEE Access*, Vol. 6, 2018, pp. 12118–12128.

[32] Alphand, O., et al. "IoTChain: A Blockchain Security Architecture for the Internet of Things," *IEEE Wireless Communications and Networking Conference (WCNC)*, 2018, pp. 1–6.

[33] Tsai, W., X. Bai, and L. Yu, "Design Issues in Permissioned Blockchains for Trusted Computing," *2017 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, San Francisco, CA, 2017, pp. 153–159.

[34] Huang, Z., et al., "A Decentralized Solution for IoT Data Trusted Exchange Based on Blockchain," *2017 3rd IEEE Int. Conf. Comput. Commun.*, 2017, pp. 1180–1184.

[35] Astaras, S., et al., "Deep Learning Analytics for IoT Security over a Configurable BigData Platform," *Proc. of the 22nd International Symposium on Wireless Personal Multimedia Communications, IEEE WPMC*, Lisbon, Portugal, November 2019.

[36] Soldatos, J., (ed.), "Security Risk Management for the Internet of Things: Technologies and Techniques for IoT Security, Privacy and Data Protection," Boston-Delft: now publishers, 2020, https://www.nowpublishers.com/article/BookDetails/9781680836820.

CHAPTER 8

IoT Standards and Ecosystems

This chapter presents some of the most popular IoT standards. Standards are extremely important for the fast penetration and wider adoption of IoT technologies and applications. They alleviate the heterogeneity of the different implementations, while boosting technological longevity. It is therefore important for IoT stakeholders to understand the main standards that they must consider adopting and deploying as part of their IoT systems. Nevertheless, the IoT standards landscape is very rich and comprises a wealth of heterogeneous standards. This chapter presents some of the most widely used standards. Moreover, it classifies the selected standards according to their functionality and role in the IoT systems development process.

Apart from IoT standards, this chapter introduces some of the most prominent IoT ecosystems where applications and services are developed, deployed, and operated. Ecosystems bring together developers, deployers, service providers, consumers, end users, and other stakeholders. They provide tools and techniques for the development, deployment, and operation of novel IoT applications. It is therefore a common practice for IoT developers and deployers to follow the practices and standards of certain ecosystems. Hence, understanding the various ecosystems can help IoT stakeholders to start and advance their IoT development efforts on the right foot.

Overall, this chapter is divided into two main parts: one devoted to the discussion of different IoT standards and one focused on IoT ecosystems.

8.1 IoT Standards

8.1.1 The Wealth of IoT Standards

The popularity of the IoT paradigm is reflected in the emergence of numerous standards, which are developed and promoted by many different standards development organizations (SDOs) (see, for example, [1, 2]). These SDOs can be classified in different ways, such as:

- *The functionality of the IoT application that they address:* There are, for example, standards dealing with IoT networking, others focused on IoT devices connectivity, and standards for IoT application development. Chapter 7 presented several IoT security standards.

- *The IoT applications areas that they target:* There are many application-specific standards, such as standards for smart cities, smart home, and healthcare applications. Chapter 10 presents some of the most popular IoT application areas. There are standards tailored to the needs of applications in these areas.
- *The SDO that produces these standards:* In most cases, SDOs produce entire sets of standards covering different functionalities and addressing different application areas. Therefore, IoT standards can be clustered into different families based on the SDO that develops them. For example, there are the Institute of Electrical and Electronics Engineers (IEEE) standards for IoT devices and applications, standards developed by the European Telecommunications Standards Institute (ETSI), and IoT-related standards from the World Wide Web Consortium (W3C).

Taking the IoT standards that are developed by the IEEE as an example, there are many IEEE IoT-related standards for smart cities, including standards for transport and healthcare. For instance, there are IEEE standards addressing the operation of wearables devices, medical devices, connectivity, and transport, as well as transfer and exchange of data from the devices to a cloud infrastructure. Here is a list of IEEE standards and initiatives for healthcare:

- *IEEE 2010-2012 Recommended Practice for Neurofeedback Systems:* This presents requirements for the documentation of neurofeedback instruments and software to provide quality and availability of information to their users.
- *IEEE P1708 Standard for Wearable Cuffless Blood Pressure Measuring Devices:* This establishes a normative definition of wearable cuffless blood pressure-measuring devices and the objective evaluation of this kind of devices.
- *IEEE P1859 Standard for Relaxor-Based Single Crystals for Transducer and Actuator Applications:* This covers the physical and electromechanical requirements for relaxor-based piezoelectric single crystals of lead magnesium niobate-lead titanate (PMN-PT) and lead zinc niobate-lead-titanate (PZN-PT) solid solutions of perovskite structure,
- *IEEE P3333.2 Three-Dimensional (3-D) Medical Modeling:* This provides routine visualization techniques for 3-D medical images, so that medical images can be visualized from a routine process.
- *IEEE 802.15.4j Physical Layer Extension:* This supports medical body area network (MBAN) services operating in the 2,360–2,400-MHz band.
- *IEEE 802.15.6 Standard for Wireless Body Area Networks:* This is a standard for short-range, low-power, and highly reliable wireless communications in, on, and around the human body.

This list includes only a small subset of IEEE standards for healthcare and life sciences. Furthermore, some of the listed standards are not among the most widely used. Nevertheless, it demonstrates the wealth of standards that can be considered when developing the different parts of an IoT system for healthcare. Similarly, there

is a wide range of IEEE standards for smart home applications, including standards that cover different aspects of smart home functionalities such as smart grid, video transmission, electric vehicles, home networking, smart metering, and more. These smart home standards cover not only applications within the home, but also their interaction with other IoT systems and devices such as vehicles and smart meters.

8.1.2 3GPP LTE Standards for IoT Networking and Connectivity

In the area of IoT systems connectivity, the Third Generation Partnership Project (3GPP) provides some of the most used connectivity options and technologies. The 3GPP provides its members with a stable environment to produce the reports and specifications that define 3GPP technologies. The three technical specification groups (TSG) in the 3GPP cover the following areas: radio access networks (RAN), services and systems aspects (SA), and core network and terminals (CT).

The main outcome of the 3GPP is the specifications of the Long-Term Evolution (LTE) standards for wireless and mobile communications, that is, the standards of the fourth generation (4G) of mobile or wireless communications [3]. LTE provides a range of characteristics that make it suitable for the connectivity of IoT devices and applications, including:

- An all-Internet Protocol (IP) architecture, which makes it ideal for IoT applications;
- Built-in security along with robust and scalable traffic management capabilities;
- Significantly more spectral efficiency than 2G or 3G. Transporting data over a 4G LTE network can be done at a much lower cost per bit. LTE is two or three times more efficient than 3G and 20 times more efficient than 2G.

4G is now mainstream in terms of supporting IoT applications. However, it has limitations in terms of supporting the exponentially proliferating IoT devices and applications. These limitations are discussed in Chapter 9, where 5G is presented as the ideal solution for IoT connectivity at a large scale. However, 5G is not yet commercially available at large scale, yet telecommunication operators are in the process of licensing spectrum, deploying 5G equipment, and conducting trials and tests.

When it comes to IoT networking, IP-based solutions offer many advantages. One of them relates to the use of IPv6 for IoT naming and addressing, which solves the ever-important problem of the identification of IoT devices. However, resource-constrained devices cannot always support a Transmission Control Protocol/Internet Protocol (TCP/IP) or User Datagram Protocol (UDP)/IP network stack. To this end, there is frequently a need for enriching their capabilities to allow them engage in end-to-end IP communications from the device to the IoT application. 4G networks facilitate all IP architectures. Nevertheless, this is also the case with other protocols, such as the low-power wide area networks (LPWAN) that are discussed next.

8.1.3 LPWAN Standards

8.1.3.1 Overview of LPWAN Technologies

4G networks and legacy wireless local area networks (i.e., Wi-Fi standardized based on the IEEE 802.11 family of standards) carry a significant portion of the IoT traffic worldwide. In recent years, a new set of standards and technologies for IoT networking and connectivity have emerged, namely, the LPWAN standards and technologies. LPWAN technologies are disrupting the IoT infrastructure landscape and hold the promise of revolutionizing the deployment of several IoT applications, notably smart city applications.

LPWAN technologies enable specialized wireless wide area networks that interconnect devices with low-bandwidth connectivity. In this way, they prioritize range and power efficiency. There are many LPWAN technologies and standards, which share the following characteristics:

- They support bidirectionality and cover larger areas than mainstream mobile networks, while consuming less power.
- They are primarily targeted to supporting IoT and machine-to-machine (M2M) applications. Their low power and long-range characteristics make them more suitable for IoT and smart city deployments, when compared to legacy mobile networking technologies.
- They feature low data transfer rates, which makes them suitable for devices that have low bandwidth requirements. LPWAN technologies support the connectivity of devices that are less bandwidth-savvy than standard home equipment.
- They are ideal for communications in industrial environments with many devices, as they are power and cost-efficient.

These characteristics make LPWAN technologies a perfect fit for IoT applications that comprise numerous IoT devices. For example, they provide a compelling value proposition for smart cities applications such as energy management and water management applications.

These advantages of LPWAN systems have led many vendors and SDOs to introduce relevant technologies.

Figure 8.1 illustrates a typical IoT architecture that leverages some LPWAN protocol for device connectivity, along with 4G connectivity for the LPWAN base stations or gateways. As depicted Figure 8.1, LPWAN and 4G can be combined in an IP-based IoT architecture. Next we present the most popular LPWAN technologies.

8.1.3.2 LoRaWAN

LoRaWAN is specified by the LoRa Alliance and supports the connectivity of wireless battery-operated objects. It operates in regional, national, and global scopes, which makes it suitable for smart city deployments of multiple scales. LoRaWAN supports secure, bidirectional communications, mobility, localization, and interoperability services [4]. Most importantly, LoRaWAN is deployed in a flexible and

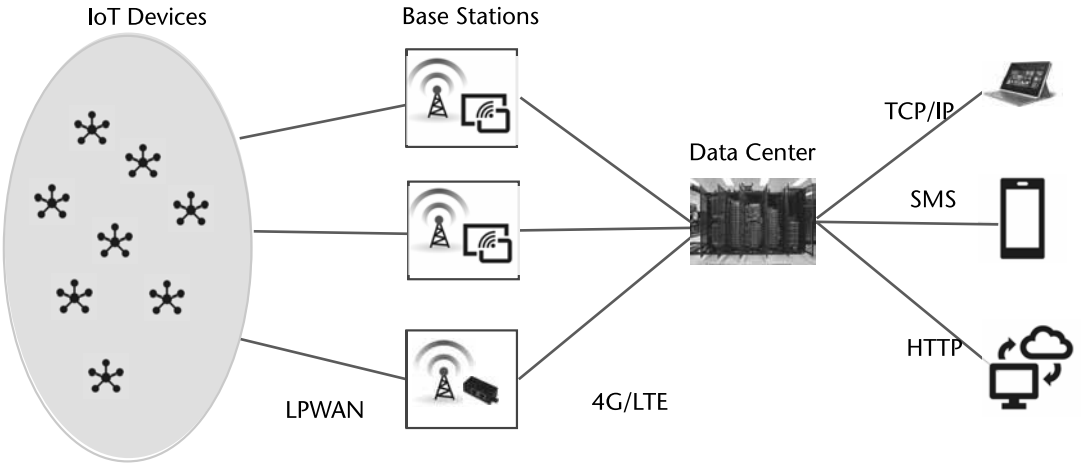

Figure 8.1 Typical IoT architecture based on LPWAN and 4G/LTE.

cost-effective way, which obviates the need for complex installations. As such, it facilitates IoT deployers (e.g., network operators, cities, plant operators) to establish enterprise-scale infrastructures and to roll out services on top of them. The deployment of a LoRaWAN network is usually based on a star-of-stars topology, which comprises gateways that transfer relay messages between connected devices and a back-end server. End devices such as sensors and smart meters connect to the gateways based on wireless connectivity using various frequency channels and adaptive data rates, which depend on communication range and the volume of exchanged data. From a network perspective, LoRaWAN is a media access control (MAC) layer infrastructure that employs its own modulation scheme. This is the reason why LoRAWAN networks are in usually described as LoRA networks. LoRA modulation is also the foundation for other LPWAN technologies such as Symphony Link.

8.1.3.3 Narrowband IoT (NB-IoT)

NB-IoT takes advantage of cellular spectrum bands to provide devices connectivity [5]. NB-IoT is a narrowband radio technology, which is standardized by the 3GPP. It is based on LTE radio specifications, can be deployed on existing LTE infrastructure, and is compatible with broadband LTE systems. It provides excellent indoor coverage at a low cost, while ensuring long battery life for the connected devices [6].

A companion 3GPP standard is LTE Cat-M (Machine), which has higher-power consumption and data rates than NB-IoT. It is being deployed ahead of NB-IoT in some regions, most prominently by U.S. operators. Some operators will deploy both technologies in parallel for different use cases, while others will rely entirely on NB-IoT and, in the future, 5G-IoT.

8.1.3.4 Ultranarrow Band (UNB)

UNB technology transmits over very narrow spectrum channels (i.e., typically <1 kHz) to achieve ultra-long-distance linking of transmitters and receivers. UNB technology is used by SigFox for its IoT deployments that include quite sophisticated base stations, which monitor the full 192-kHz spectrum and look for UNB signals to demodulate [7]. Sigfox's deployments are based on a cellular-like UNB system. The company targets the development of a global network outside the licensed spectrum, in collaboration with a network of partners worldwide. SigFox's prominent position in the UNB market is the reason why SigFox is commonly used as synonymous to UNB.

8.1.3.5 Wi-Fi HaLow

Wi-Fi HaLow is specified by the Wireless Broadband Alliance (WBA) in order to operate in the 1-GHz Industrial, Scientific and Medical (ISM) band. It is based on the IEEE 802.11ah protocol and offers longer-range, lower-power operation, yet with lower throughput when compared to other Wi-Fi technologies. As such, it is destined to support IoT devices deployed in larger areas.

8.1.3.6 Discussion of LPWAN Technologies and Standards

There are also other, less widely used technologies such as Haystack and Random Phase Multiple Access (RPMA). The above-listed LPWAN technologies are also able to support security-sensitive applications, such as those involving critical infrastructures (e.g., energy, transport) or exchange of personal data (e.g., healthcare), thanks to their multilevel encryption schemes. Specifically, LoRaWAN offers encryption and security functionalities at the network, application, and device levels.

LPWAN technologies are not destined to replace other IoT-related networking technologies and standards. Rather, they are an excellent complement to other types of networks (e.g., 3G/4G satellite, as shown in Figure 8.1). This facilitates the interconnection of assets and devices that are dispersed across a wide area, including devices that are not properly supported by preexisting cellular networking technologies. For example, NB-IoT and LTE are both specified by the 3GPP to cover different networking needs in the LTE spectrum. Overall, LPWAN technologies complement existing IoT networking infrastructures and facilitate the rapid and cost-effective deployment of various IoT applications that are very costly when deployed over legacy networks.

In a nutshell, LPWAN technologies provide the following advantages:

- *Cost-effectiveness:* LPWAN solutions are cost-effective for various reasons. First, they are usually straightforward to install, as they can use battery-operated sensors. Battery-operated sensors last for several years, which alleviates the need for a power source, as is the norm with technologies such as LTE and Wi-Fi. Second, LPWAN technologies provide excellent indoor coverage and penetration, which enables devices' deployment both indoors and outdoors within a range of several kilometers. Hence, LPWAN deployments

do not require complex studies in terms of coverage analysis and support for indoor applications. Third, some LPWAN technologies (e.g., LoRaWAN, UNB) can be deployed in the ISM radio band, which entails no spectrum costs. Overall, LPWANs are cost-effective not only in terms of their deployment, but also in their operation. Finally, there are already several providers of LPWAN (e.g., SigFox, NB-IoT, LoRaWAN) services, which boost competition and lowers prices for LoRA services.

- *Geolocation features and Location-as-a-Service (LaaS):* LPWANs enable geolocation applications since they support location functionalities without any need for additional power. Furthermore, based on their indoor coverage capabilities they can support LaaS functionalities in smart environments such as warehouses, airports, and smart buildings. LaaS is a key enabler for a wide range of added-value applications in areas such as asset management, supply chain management, security, and safety.
- *Expanding ecosystem:* The most prominent LPWAN technologies (e.g.., LoRaWAN, NB-IoT) are associated with large ecosystems. For example, LoRaWAN is an open specification by the LoRa Alliance, which is supported by more than 400 members, including some of the most prominent vendors and network operators, as well as start-ups. These industrial organizations form a growing ecosystem, which provides a strong push to the technology through deploying LoRaWAN infrastructures and developing relevant applications. Moreover, the alliance provides a host of supporting services such as interoperability certification for IoT products and services.
- *Compliance with legacy systems:* LPWAN technologies can achieve excellent penetration in indoor environments and device-saturated urban areas. This facilitates the interconnection and deployment of legacy infrastructures and devices as part of LPWAN deployments. The ability to reuse existing infrastructures and devices in the scope of IoT applications boosts the cost-effectiveness of LPWAN technologies.
- *Security:* LPWAN gateways and specifications come with a range of built-in security features such as the Advanced Encryption Standard (AES)-128 encryption, which make them appropriate for applications that require strong security.

LPWAN technologies are already available for both public and private deployments within smart cities and other IoT applications. Specifically, modern cities invest in their own municipal network to accommodate multiple applications such as waste management, smart parking services, street lighting, field service engineering for buildings, and air quality management. These applications are supported by the integration of devices with the LPWAN infrastructures that they deploy.

8.1.4 IETF Constrained Application Protocol (CoAP)

The CoAP is a specialized Web Transfer Protocol for use with constrained IoT nodes and networks. It is designed for M2M applications such as smart energy and

building automation. CoAP enabled devices expose developer-friendly REST APIs, which facilitate developers to write applications over them [8]. There is already rich language support for writing applications over CoAP devices, including C/C++, Java, and Python. CoAP is a specification of the Internet Engineering Task Force (IETF), which is described in the Request for Comments (RFC) 7252.

CoAP enables the popular REST (Representational State Transfer) protocol for small devices. With CoAP, servers make resources available under a Uniform Resource Locator (URL), while clients access these resources using GET, PUT, POST, DELETE, and other commands of the popular Hypertext Transfer Protocol (HTTP). However, CoAP is not like most HTTP-based protocols. For instance, it operates over UDP and does not implement complex congestion control mechanism as is the norm in TCP. Moreover, it is based on a REST architecture and a general design for accessing resources on the internet. It does not provide support for the full range of HTTP methods. However, it supports IP multicast to enable point-to-multipoint communications in the scope of IoT applications. CoAP provides better reliability than standard UDP based on a retransmission mechanism. Overall, CoAP is more lightweight than HTTP and hence suitable for supporting communications between resource-constrained devices on a large scale. Figure 8.2 presents the HTTP and CoAP protocol stacks, where the above-listed differences are illustrated. Even though the two protocols have several differences and a different set of supported methods, it is possible to map HTTP messages into CoAP as presented in [9].

CoAP is developer friendly as it is used based on techniques and APIs that are very popular among the millions of web developers (i.e., REST, WebAPI). CoAP has been made to support billions of nodes, including inexpensive ones. For example, it can work on microcontrollers with as low as 10 KB of RAM and 100 KB of code space (i.e., devices that are extremely constrained in terms of computing resources). Furthermore, it is well-designed and secure, offering the following characteristics:

- It deals with difficult resource and traffic management issues, such as congestion control. However, as already outlined, it provides more lightweight congestion control than TCP and HTTP.

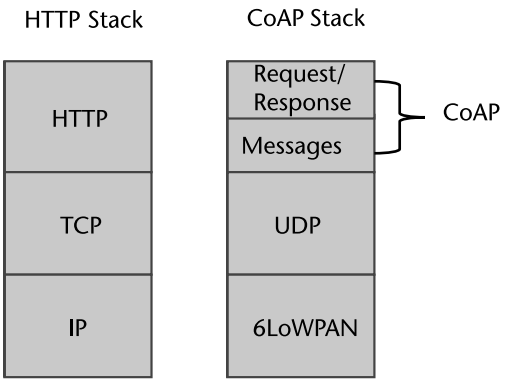

Figure 8.2 HTTP stack versus CoAP stack.

- It provides strong security, as CoAP's default choice of Datagram Transport Layer Security (DTLS) parameters is equivalent to 3,072-bit RSA[1] keys.
- It runs fine even on the smallest nodes (i.e., central processing unit (CPU) constrained nodes).

CoAP is integration-ready, as HTTP and CoAP share the REST model. It can easily be connected using application-agnostic cross-protocol proxies, while being transparent to web clients. Moreover, it offers flexibility in data models since it integrates and serializes information based on eXtensible Markup Language (XML), JavaScript Object Notation (JSON), Concise Binary Object Representation (CBOR), and other popular data serialization formats.

It uses minimal resources and provides resource efficiency at the device and the network levels. It operates based on UDP over IP, including a 4-byte fixed header and a compact encoding of options. In this way, it enables small messages that cause little to no fragmentation on the link layer. Moreover, it supports integrated discovery based on a CoAP resource directory, which enables discovery of resources based on different properties of the IoT nodes.

The rising popularity of CoAP is reflected in a proliferating number of implementations of the protocol in different programming languages. This provides versatility for developers, who can program CoAP in many different platforms. A list of relevant implementations is:

- *Erbium:* This is a REST Engine and CoAP Implementation for Contiki, a widely used operating system for constrained nodes.
- *Libcoap:* This is a C implementation of CoAP. It can be used both on constrained devices and on a larger POSIX system. It can be also ported to TinyOS and RIOT.
- *Tinydtls:* This is a Tiny implementation of DTLS for Class 1 devices.
- *SMCP:* This is a small but capable C-based CoAP stack suitable for embedded environments. It supports observation, asynchronous responses to requests, and more.
- *Microcoap:* This is a C implementation that can be compiled for both Arduino and POSIX environments.
- *Cantcoap:* This is a C implementation that focuses on decoding and encoding, leaving the actual protocol to the application. In this implementation, the user is expected to deal with retransmissions, timeouts, and message ID matching himself or herself.
- *Lobaro CoAP:* This is a C implementation for constrained devices that covers base CoAP, Observe, and Block.

1. The RSA encryption acronym stands for Rivest, Shamir, and Adelman, the inventors of the encryption technique.

- *Wakaama:* This covers the LWM2M Protocol, CoAP, and DTLS layers of the LWM2M protocol stack for all three logical components: LWM2M Client, LWM2M Server, and LWM2M Bootstrap Server.

In addition to the previous implementations of CoAP on different types of devices, there are also server-side implementations. The latter provide implementations of the server parts of the CoAP protocol stack. Server-side implementations are available in different languages, including:

- Java implementations, such as Californium (Cf, https://eclipse.org/californium), nCoAP (https://github.com/okleine/nCoAP) that uses the Netty NIO client server framework and Leshan (http://eclipse.org/leshan), which is an OMA LWM2M server-side implementation, on top of Cf;
- C and C# implementations, such as Libcoap (https://libcoap.net), CoAP.NET, which is an implementation in C#, providing CoAP-based services to .NET applications and CoAPSharp, which is a lightweight library for building CoAP-enabled sensors and machines that also works with the Microsoft .NET Micro Framework;
- Erlang and Javascript, including the gen_coap, which is a pure Erlang implementation of a generic CoAP client and server and node-coap which is a client and server library for CoAP modelled after the http module.

Server-side implementations of CoAP are also available for Python and Ruby, as well as for smartphones and browsers.

CoAP is one of the most popular protocols that enables access to IoT devices. However, there are also alternatives to CoAP [10], such as the MQTT technology discussed in Chapter 2 and oneM2M, presented next.

8.1.5 oneM2M

8.1.5.1 oneM2M Overview

oneM2M is a standard developed in 2015 managed by a partnership of regional international standards industry organizations in the telecommunications industry. oneM2M provides a common service layer that sits between applications and connectivity transport. It offers functions that are commonly needed by IoT applications across different industry segments. Those functions are exposed to applications via RESTful APIs.

oneM2M deals with the interaction of an application with different network and services infrastructures, which are represented by application domains, network domains, and M2M service domains. Clients interact with M2M applications through a set of service enablers that abstract network and services functionalities.

oneM2M standards comprise a horizontal platform architecture that fits within a three-layer model comprising applications, middleware services, and networks. oneM2M's connectivity standards permit applications that are hosted on connected machines and devices, enterprise systems, and mobile devices to communicate

with each other in an efficient, secure manner. The oneM2M horizontal platform is scalable as the common service elements are deployed on hosts, at the proximal network edge or within the enterprise cloud.

Connectivity services provide capabilities that allow for efficient communication between application endpoints. They provide interworking mechanisms that adjust the underlying network (e.g., mobile, wireless) QoS to meet the needs of current application data exchange. Currently, the oneM2M service layer can use HTTP, CoAP, MQTT, and WebSockets for connectivity transports. Data distribution services (DDS) are also being explored as another option for providing real-time connectivity between the oneM2M service layer entities.

The typical usage of oneM2M includes registration and subscription of devices and applications, service charging and accounting, management of application and devices, and monitoring.

oneM2M has various commercial deployments in home-automation applications. It is suitable for large-scale consumer IoT applications. oneM2M is also actively used in deployments in other IoT application domains, including telematics and intelligent transportation, home automation, utilities, healthcare, smart cities, and industrial automation. In all these domains, oneM2M provides semantic enablers in the scope of its distributed data interoperability and management layer. As a key part of the telecommunication industry's existing initiatives and its new 5G initiative, oneM2M provides enablers that focus on the connection between device and the cellular network.

oneM2M's horizontal layer involves the following entities and stakeholders:

- M2M applications providers, who run individual M2M services. The customer of an M2M service is typically the owner of the device through which the service is delivered.
- M2M service providers, which hosts several M2M applications on their platforms.
- Wide area transport network operators, which offer networking services to the M2M service provider.
- End user, who owns or operates the device or gateway.

8.1.5.2 oneM2M's Vision

The overall vision of oneM2M standard as provided by ETSI, includes the following aspects:

- Lower capital expenses (CAPEX) and operational expenses (OPEX) for M2M services;
- Accelerated time to market for M2M services, based on reduced investment barriers;
- Interoperability across solutions, which reduce costs using simplified M2M services for both users and application developers.

The previously described targets are achieved in oneM2M through:

- Providing standards-based APIs and protocols, which reduce complexity and minimize initial investments;
- Improved network efficiency based on a single (network-independent) service layer for different vertical applications;
- Standardized interfaces and protocols in the scope of a multivendor ecosystem;
- Hiding the complexity of the underlying networking infrastructure, while facilitating developers to foster innovation of new services.

8.1.5.3 oneM2M Services

From a developer's perspective, oneM2M is a software layer between M2M applications and communication. It also includes hardware and software that provide transport layer services. oneM2M implementations are typically IP-based and provide functions needed across M2M applications by different industry segments. Functionalities are exposed to applications via IT-friendly APIs, which enables distributed intelligence (i.e., intelligence across devices, gateways, and cloud apps).

The oneM2M standardization process is driven by use cases in different verticals (e.g., automotive, energy, e-health), which have provided requirements in terms of security and privacy, device management, data exchange, and interworking between devices. Based on these requirements, oneM2M has specified a range of APIs and protocols, based on RESTful technologies and interfaces. Finally, the standards development process includes testing of the protocols and the APIs, as well as interoperability testing.

A set of common services are defined as part of a oneM2M implementation. These include registration and discovery services, security services, group management services, data management services, subscription and notification services (i.e., publish subscribe services), device management services, services management services, communication management services, network services exposure, location services, and charging and accounting services.

8.1.6 IETF 6LoWPAN

6LoWPAN (www.6lowpan.org) is a simple low throughput wireless network comprising low-cost and low-power IoT devices such as sensor networks. It supports common topologies such as star, mesh, and combinations of star and mesh. LoWPAN networks are typically used for supporting:

- Networking transducers (e.g., sensing and actuation, smart sensors);
- Simple control applications such as control in smart home environments;

- Complex networked controls such as light, switch, and motion sensors. LoWPANs are a reality and are already deployed in various applications (e.g., smart home).

In both star and peer-to-peer topologies, there are devices with reduced capability sets, which are communicating and connecting to the network through full functional devices (i.e., devices with more advanced computational capabilities).

6LoWPAN operates on the following assumptions:

- All devices or nodes conform to IEEE 802.15.4 standard.
- Devices send small amounts of data.
- Devices are typically constrained in terms of computing, power, cost, memory, storage (i.e., they are IoT devices).

Some of the main characteristics of the technology include:

- Small packet size;
- 16-bit short or IEEE 64-bit extended MAC addresses;
- Operation based on low bandwidth (250/40/20 kbps);
- Support for various topologies including star and mesh;
- Low-power operation, since devices are typically battery-operated;
- Devices are relatively low-cost;
- Networks are ad hoc and devices have limited accessibility and user interfaces.

The overall settings where 6LoWPAN operate are inherently unreliable due to nature of devices in the wireless medium. Popular applications for 6LoWPAN include equipment health monitoring, environment monitoring, security, smart home applications, and building automation.

A very typical configuration involves the interconnection of a 6LoWPAN network or domain with an IPv6 domain or network. This integration with IPv6 facilitates the use of IPv6 as an identification technique or mechanism for 6LoWPAN.

8.1.7 HyperCat

As already mentioned, interoperability across different IoT systems is crucial for many IoT applications. Motivated by the need for interoperability, the HyperCat Consortium created a lightweight standard for the interoperable exchange of data and services across diverse IoT systems. The HyperCat standard is driving secure and interoperable IoT deployments for industry and cities. In practice, the HyperCat specification:

- Allows IoT clients to discover information about IoT assets over the web, thus boosting discovery;

- Enables developers to write applications that will work across many servers, breaking down the walls between vertical silos;
- Enables interoperability across vertical applications.

The standard is a lightweight JSON-based hypermedia catalog format for exposing collections of URLs with information about IoT assets over the web. Its main characteristics are as follows:

- It is extremely simple and uses simple and well-known technologies such as HTTPS, REST, and JSON.
- Each HyperCat catalog may expose any number of URIs, each with any number of resource description framework-like (Resource Description Format (RDF)-like) triple statements about it.
- HyperCat allows a server to provide a set of resources to a client, each with a set of semantic annotations.
- Implementers are free to choose or invent any set of annotations to suit their needs.
- A set of best practices and tools has been developed and deployed in various cities, notably in the United Kingdom.

Successful deployments of HyperCat in the United Kingdom have given rise to its adoption in other countries, such as Australia.

8.1.8 Open Geospatial Consortium (OGC) Standards

The OGC is a worldwide community committed to advancing geospatial location information and services as a vital force for progress. For over a decade, OGC has produced a range of standards for IoT applications involving sensors and using geolocation information. Most OGC standards specify interfaces and encodings between different components and services that comprise IoT-based multisensor applications. Hence, products and services implementing OGC standards boost interoperability across independently developed components. Likewise, these standards enable the plug-and-play composition of the IoT components.

SensorML is one of most prominent standards of the Open Geospatial Consortium. It provides standard models and an XML encoding for describing any process. Specifically, it enables a description of:

- The process of measurement by sensors;
- Instructions for deriving higher-level information from observations.

SensorML processes are discoverable and executable. They are defined in terms of inputs, outputs, parameters, methods, and relevant metadata. Moreover, a SensorML model can be classified either as a detector or as a sensors-based process that converts real phenomena to data.

In practice, SensorML is an XML schema. It defines the geometric, dynamic, and observational characteristics of a sensor, while also providing or describing functional models for sensors. A SensorML representation includes a detailed description of sensor hardware and separates the sensor from its associated platform(s) and target(s). SensorML descriptions are used in different ways, including:

- As a means for describing sensors' information to support data discovery;
- As a way for processing and analysis of sensor measurements;
- As a means of deriving the geolocation of measured data;
- As a way for accessing performance characteristics (e.g., accuracy, threshold) of the sensor process;
- As a means of defining and accessing fundamental properties and assumptions regarding the sensor.

The description of a single sensor with SensorML is the simplest use case for this markup language. SensorML provides the means for combining multiple sensor descriptions in an integrated component. A SensorML description comprises various sections, which are typically dedicated to the description of the following entities:

- System description;
- Observed properties;
- Outputs or quantities;
- System location;
- System components;
- Connections between components and system output.

SensorML is not confined to modeling a single sensor. Rather it is destined to model entire sensing processes. In SensorML, everything is modeled as a process, based on a process model that defines atomic process modules. The SensorML process model comprises five sections, namely: metadata, inputs, outputs, parameters, and method. A process defines a process chain, which includes metadata, inputs, outputs, and parameters, processes (ProcessModel, Process), and data sources. It also includes a description of the connections between processes and between processes and data. A sensor system is accordingly defined as a collection of related and interconnected processes.

OGC provides also standards for the description of transducers and transducers' processes (i.e., TransducerML), as well as standards for expressing observations and measurements (i.e., OGC O&M). These can be combined with SensorML to support the description and operation of more sophisticated, yet interoperable IoT systems.

8.1.9 Standards-Based IoT Architectures

In earlier chapters, we outlined standards-based architectures for IoT systems, such as the Industrial Internet Reference Architecture (IIRA) and the OpenFog Reference Architecture of the Industrial Internet Consortium. These architectures provide structuring principles and building blocks for developing, deploying, and operating IoT systems. In the area of smart manufacturing and the IIoT, a Reference Architecture (RA) for Industry 4.0 systems has been also introduced, namely, the Reference Architecture Model Industry 4.0 (RAMI 4.0). RAMI 4.0 introduces structuring concepts and provides vocabulary for understanding Industry 4.0 systems and their deployment. RAMI describes the structure and main elements of Industry 4.0 system by means of a 3-D layered model (Figure 8.3). The three layers of the 3-D model correspond to:

- *The Architecture axis (Layers):* This comprises six different layers indicating functionalities at different granularities of the system, from the asset to the business level.
- *The Process axis (Value Stream):* This illustrates the stages of an asset's life cycle, along with a corresponding value creation process based on the IEC 62890 standard.
- *The Hierarchy axis (Hierarchy levels):* This presents the breakdown structure of assembled components based on a taxonomy that starts from the product and goes up to the connected smart factory.

Likewise, the architecture layers of RAMI 4.0 include:

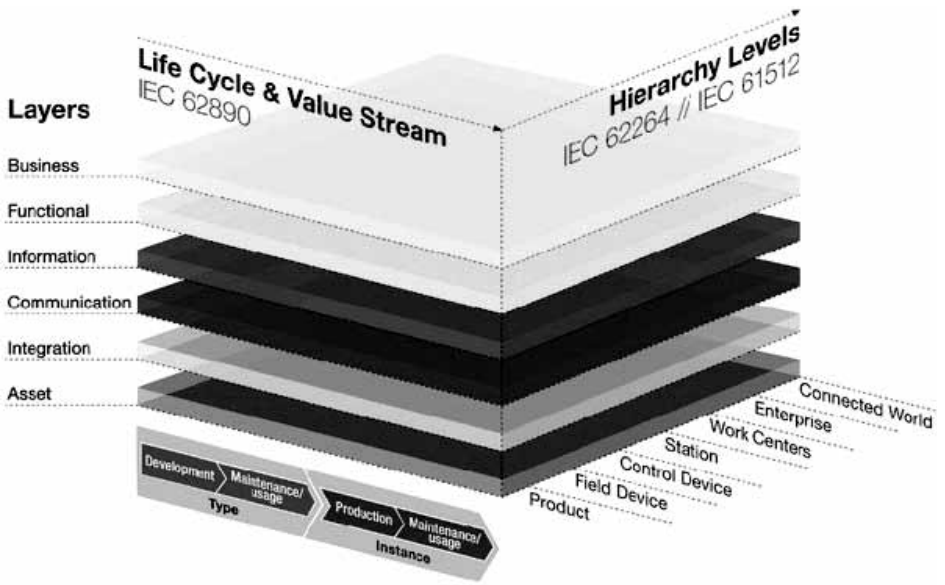

Figure 8.3 Reference Architecture Model Industry 4.0 [11]. (© Plattform Industrie 4.0 & ZVEI Zentralverband Elektrotechnik und Elektronikindustrie e.V. Reprint with Permission.)

- *The asset layer:* This describes physical systems and components (e.g., machines, motors, software applications, spare parts).
- *The integration layer:* This links the physical and digital or cyber worlds based on components such as drivers and middleware.
- *The communication layer:* This deals with communications between the integration and information layers. It employs network protocols (e.g., TCP/IP, HTTP, File Transfer Protocol (FTP)) over local area networks (LANs) and wide area networks (WANs), including wireless networks.
- *The information layer:* This provides digital information about sales, purchase orders, suppliers, and locations along with information on materials, machines, and components that support the production.
- *The functional layer:* This comprises production rules, actions, processing, and system control.
- *The business layer:* This is associated with the business strategy, the business environment, and business goals of the enterprise, including promotions, offers, pricing models, and cost analysis.

8.2 IoT Ecosystems

8.2.1 Introduction to IoT Ecosystems

A business ecosystem is defined as a strategic planning model. It has been a popular term since the early days of the development of information technology, whereby a network of suppliers, distributors, competitors, and customers all work through competition and cooperation to advance sales of products.

In the case of IoT business ecosystems, a wide range of stakeholders are participating in the production and sales of products and services. These stakeholders participate in IoT-based value chains or impact the environment where IoT businesses operate. The stakeholders of an IoT ecosystem include:

- Regulators or government;
- Telecom operators;
- Customers and consumers;
- Standards bodies and industry groups;
- Hardware vendors;
- Software vendors;
- Middleware vendors;
- IT service vendors;
- Cloud service providers;
- Innovators and entrepreneurs.

Moreover, IoT products span a wide range of vertical markets, which are linked to the IoT business ecosystem. Prominent market segments include:

- Consumer goods (e.g., smartphones, smart homes, smart cars, appliances);
- E-health areas, including fitness, bioelectronics, and healthcare;
- Smart transportation;
- Energy distribution (e.g., smart grid);
- Smart cities;
- Distribution and logistics;
- Public safety;
- Smart industry and manufacturing;
- Agriculture and natural resources management.

The list is not exhaustive as there are more IoT market segments. Furthermore, it is likely that additional markets will also emerge in coming years.

Standardization bodies and SDOs are also part of the IoT ecosystem, as they are in charge of standards development. Prominent SDOs in the IoT space include:

- The IEEE: www.ieee.org;
- The International Electrotechnical Commission (IEC): www.iec.ch;
- The International Organization of Standardization (ISO): www.iso.org;
- The International Society of Automation (ISA): www.isa.org;
- The International Telecommunication Union (ITU): www.itu.int;
- The Internet Engineering Task Force (IETF): www.ietf.org;
- The W3C: http://www.w3.org;
- The ETSI: http://www.etsi.org.

8.2.2 Data-Centric IoT Products

One specific category of IoT products concerns data producers and data-centric products. These products are typically internet-connected devices, which are used in the scope of different IoT settings and applications such as smart cities (e.g., sensors in the road surface), wearable activity trackers (such as Fitbit), environment-aware thermostats (such as Nest), and a Bluetooth dongle that can read vehicle data and can be deployed in connected car applications. These devices come with added value functionalities, which are provided as part of their IoT ecosystem.

8.2.3 Vendors' Ecosystems

Major vendors of IoT infrastructures and services have established IoT ecosystems around their platforms. This is the case with Microsoft, Amazon, Xively, and other vendors presented in Chapter 4. Developers, solution providers, and innovators

around these ecosystems take advantage of vendors' platforms and tools, in their efforts to build the next generation of IoT products and services.

8.3 Summary

Earlier chapters of the book illustrated that IoT is not a single technology, but rather the outcome of the integration of a pool of different technologies. These technologies cover a wide range of functionalities such as networking, connectivity of devices, and M2M interactions. This is the main reason why there are many and different IoT standards. This chapter attempted to shed light on some of the most prominent IoT standards, including both legacy and emerging standards. Specifically, this chapter presented popular standards for IoT networking, connectivity, architecture development, interoperability, and description of IoT systems. The presented standards have been developed by different SDOs and serve various purposes in the context of IoT systems development. This chapter did not present standards for IoT security, as these were discussed in Chapter 7. Overall, the presented list of standards can provide guidance to practitioners seeking to understand the complex IoT standardization landscape. They could also help them to accelerate the process of exploring and selecting the standards that they must adopt as part of their IoT development and deployment endeavor.

Moving a step beyond established IoT standards, Chapter 9 presents an outlook for the evolution of IoT systems. This outlook is based on the introduction of three of the main technologies that will shape the future of IoT systems, including 5G networks, distributed ledger technologies, and AI.

References

[1] ETSI Technical Committee Smart Machine-to-Machine Communications (SmartM2M), *SmartM2M; Security; Standards Landscape and Best Practices*, ETSI Technical Report, ETSI TR 103 533 V1.1.1 (2019-08), 2019, https://www.etsi.org/deliver/etsi_tr/103500_103599/103533/01.01.01_60/tr_103533v010101p.pdf

[2] Alliance for Internet of Things Innovation (AIOTI), AIOTI WG03 on IoT Standardisation Reports and Deliverables, May 3, 2020, https://aioti.eu/aioti-wg03-reports-on-iot-standards/.

[3] Khan, A. H., et al., "4G as a Next Generation Wireless Network," *2009 International Conference on Future Computer and Communication*, Kuala Lumpar, 2009, pp. 334–338.

[4] de Carvalho Silva, J., et al., "LoRaWAN — A Low Power WAN Protocol for Internet of Things: A Review and Opportunities," *2017 2nd International Multidisciplinary Conference on Computer and Energy Science (SpliTech), Split*, 2017, pp. 1–6.

[5] Zayas, A. D., and P. Merino, "The 3GPP NB-IoT System Architecture for the Internet of Things," *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*, Paris, 2017, pp. 277–282.

[6] Chen, M., et al., "Narrow Band Internet of Things," *IEEE Access*, Vol. 5, 2017, pp. 20557–20577.

[7] Anteur, M., et al., "Ultra Narrow Band Technique for Low Power Wide Area Communications," *2015 IEEE Global Communications Conference (GLOBECOM)*, San Diego, CA, 2015, pp. 1–6.

[8] Iglesias-Urkia, M., et al., "Analysis of CoAP Implementations for Industrial Internet of Things: A Survey," *J. Ambient Intell. Human Comput.*, Vol. 10, 2019, pp. 2505–2518.

[9] Castellani, A., et al., "Guidelines for Mapping Implementations: HTTP to the Constrained Application Protocol (CoAP)," Internet Engineering Task Force (IETF), Request for Comments (RFC) 8075, February 2017, https://www.rfc-editor.org/rfc/rfc8075.html.

[10] Gündogan, C., et al., "NDN, CoAP, and MQTT: A Comparative Measurement Study in the IoT," *Proc. of 5th ACM Conference on Information-Centric Networking (ICN '18)*, Association for Computing Machinery, New York, NY, 2018, pp. 159–171.

[11] Schweichhart, K., "Reference Architectural Model Industries 4.0 - An Introduction," Deutsche Telekom, April 2016, https://ec.europa.eu/futurium/en/system/files/ged/a2-schweichhart-reference_architectural_model_industrie_4.0_rami_4.0.pdf

CHAPTER 9

Emerging IoT Technologies

Previous chapters have presented some of the mainstream IoT technologies that are commonly part of state-of-the-art systems and services. These included several legacy IoT technologies such as WSN and RFID, as well as data mining and machine learning techniques for IoT datasets. In this chapter, we introduce emerging technologies that are likely to have significant impact on future IoT systems and services. These are not widely deployed technologies that already support innovative IoT solutions. Rather, they are mostly promising technologies that are currently deployed in pilots, trials, and prototypes. Specifically, this chapter presents three main technological trends that will have significant impact on IoT:

- *The fifth generation of mobile communications (5G):* This is destined to support future IoT applications such as autonomous driving and fully automated industrial plants.
- *Blockchain technology:* This holds the promise to decentralize the state-of-the-art edge or cloud paradigm of IoT deployments towards increased security and scalability.
- *Leading edge AI technologies:* The chapter introduces AI technologies that boost the transparency, trustworthiness, automation, and efficiency of future IoT systems.

9.1 5G Networking

9.1.1 5G for IoT: The Motivation

Chapter 1 explained that IoT systems can leverage any available networking infrastructure to ensure the connectivity of devices and applications. Furthermore, in Chapter 8, we presented some of the main IoT networking technologies, including some of the most used ones in IoT deployments. For instance, a great deal of IoT traffic is carried over Wi-Fi networks, while most smartphones connect to the internet using mobile technologies of the third generation (3G) and the fourth generation (4G). Nevertheless, these networking technologies have still limitations when it comes to supporting certain types of emerging IoT applications, notably applications featuring very high data rates and involving real-time interactions between users, systems, and devices. A prominent example of such applications is found

in the emerging wave of connected and autonomous vehicles applications [1]. For instance, an autonomous car processes information from hundreds of data sources in order to understand the driving context in real time. Moreover, it takes tens of decisions per second regarding optimal and safe driving. Hence, autonomous vehicles must process very large volumes of data per second in device-saturated environments. Conventional 3G/4G networking falls short when it comes to supporting autonomous driving at scale.

The networking community is currently working on the next generation of mobile communications (i.e., 5G, which is destined to alleviate the above-listed limitations and support emerging IoT applications such as connected and autonomous transport [2, 3]). The motivation behind 5G is manifold:

- *Explosion of networked data volumes:* Today we are witnessing an explosive growth of mobile traffic, which far exceeds the capacity foreseen in 3G and 4G networks. IoT is one of the main drivers behind this explosion of network traffic, as the communicating machines (e.g., M2M communications, CPS systems) contribute unprecedented amounts of traffic, which have not been considered in the scope of 3G/4G developments.
- *Proliferation of IoT devices:* The IoT revolution is driven by an explosion of the number of internet-connected devices, which requires new networking infrastructures that could handle their interactions. IoT devices pose new requirements in terms of the management and efficient use of the wireless spectrum as the density of devices is increasing in a rapid pace.
- *New business requirements:* Emerging IoT applications (e.g., connected and self-driving cars) require very low latency and ask for novel networking infrastructures that provide exceptional QoS. In coming years, many IoT applications will have similar low-latency requirements. This has motivated the development of 5G networks.

9.1.2 Introducing 5G

5G is the next generation mobile broadband technology, which will significantly enhance the features and capabilities of previous generations of mobile communications. It is still in the early stages of development, yet advancing rapidly. Most telecommunications operators in the United States and the European Union are already running 5G technology pilots, while massive commercial rollouts are planned during the next couple of years. 5G has the following revolutionary characteristics:

- It is capable of handling 10,000 times more call and data traffic than existing 3G or 4G networks.
- Data download speeds on 5G networks are several hundred times higher than 4G.
- It changes the means of using cell phones with very high bandwidth.

Based on these features, 5G is expected to be a catalyst for the accelerated and wider adoption of certain types of emerging IoT applications. From a functional perspective, 5G enables the following scenarios:

- *Fast browsing:* 5G enables fast browsing of the internet for virtually all available applications. It provides very high bit rates and very low latency.
- *High-speed networked operations in crowded environments:* 5G delivers very good QoS even in the scope of crowded areas, such as a concert or a football stadium.
- *Communication-intensive experiences:* 5G enables communications-intensive experiences such as video playbacks or immersive experiences that follow the users. Such experiences are empowered by 5G's unique accessibility and mobility features.
- *Real-time roaming with high reliability:* 5G comes with excellent reliability features that enable real-time roaming applications with very high reliability.
- *Seamless communications between heterogeneous devices in different locations:* 5G enables seamless communications between heterogeneous things that reside in different places and connect through diverse networks all around the globe. This is largely because 5G technology provides global network coverage.

Overall, 5G represents a revolution and an evolution from existing technologies (3G, 4G, Wi-Fi) all at the same time. Specifically, 5G is revolutionary in terms of the deployment of novel technologies such as massive multiple-in, multiple-out (MIMO) wireless networks, ultra-dense networks, moving networks, and higher-frequency communications, as a means of successfully responding to traffic explosion. It is also an evolution, as it extends these technologies in support of novel applications. Furthermore, it provides some new complementary technologies such as ultra-reliable communications.

9.1.3 5G Network Functions Virtualization (NFV) for IoT Systems

5G leverages the concept of NFV in order to provide ubiquitous and global coverage for all possible IoT systems [4]. NFV decouples network functions from the low-level proprietary functionalities of devices and appliances. Specifically, it abstracts hardware appliances and provides access to the corresponding virtualized abstractions in the scope of virtual machines. Virtual machines run over the network hardware infrastructure and include routers, switches, servers, and cloud computing systems. Based on this concept, NFVs can abstract not only IoT devices, but also the data plane and control plane of different networking infrastructures.

NFV architectures have their roots in IT virtualization and are used to virtualize entire classes of network node functions. NFV functions are the building blocks that can be interconnected or chained together towards creating end-to-end communication services. Typical examples of NFV network functions include Domain

Name Service (DNS), Network Address Translation (NAT), firewalls, caching, and network security functions.

NFV architectures are very relevant for complex IoT systems and applications. The latter comprise complex physical infrastructures and heterogeneous network resources. This is the rationale behind the emergence of many NFV architectures for IoT systems.

9.1.4 Deploying 5G for IoT

5G is the way to go in order to support the rapidly proliferating number of IoT applications. 4G is already being deployed in many countries, yet is still not enough for some of the emerging IoT applications, such as autonomous cars, smart transportation, and tactile internet applications. 5G will alleviate the limitations of existing infrastructures for such applications through handling 1,000 times more mobile data than today's cellular systems. It will also provide increased data rates and reduced end-to-end latency. Furthermore, it will provide improved coverage towards a global IoT.

5G will also provide a unified framework for seamless connectivity across multiple IoT technologies (e.g., RFID, short-range communications, ultrawideband (UWB), Bluetooth), which are already used in the scope of IoT applications. Smart cities represent a prominent example where such a unified framework is required, especially when considering emerging integrated IoT applications that interconnect multiple legacy IoT deployments (e.g., traffic management and smart security systems).

IoT will leverage the enhanced reliability, the low-level latency, and the spectral efficiency of 5G in the scope of device-to-device (D2D) communication applications such as M2M and vehicle-to-vehicle (V2V) applications. At the same time, IoT/M2M applications will greatly benefit from massive machine communications in 5G. The latter will be empowered by 5G's ability to provide connectivity to large numbers of low-power devices and to support low-latency data transfers between them. Figure 9.1 presents an end-to-end connectivity scenario in a 5G context. It involves the interconnection and use of terminal devices (including IoT devices) based on the edge or cloud paradigm, leveraging on a variety of transport and access networks. 5G resources can be then orchestrated to construct proper slices that support different IoT applications. Slices are tailored to the peculiar network requirements of various IoT applications in sectors such as transport (e.g., autonomous driving), energy (e.g., smart grids), and industry (e.g., smart factory and Industry 4.0 applications).

9.1.5 5G Deployment Roadmap

At the time of this writing, 5G had entered the phase of enterprise deployment. There are regular announcements about 5G rollouts in various cities and regions all around the globe. Moreover, smartphone vendors are gradually supporting 5G in their products, while incumbent telecommunications operators are preparing for

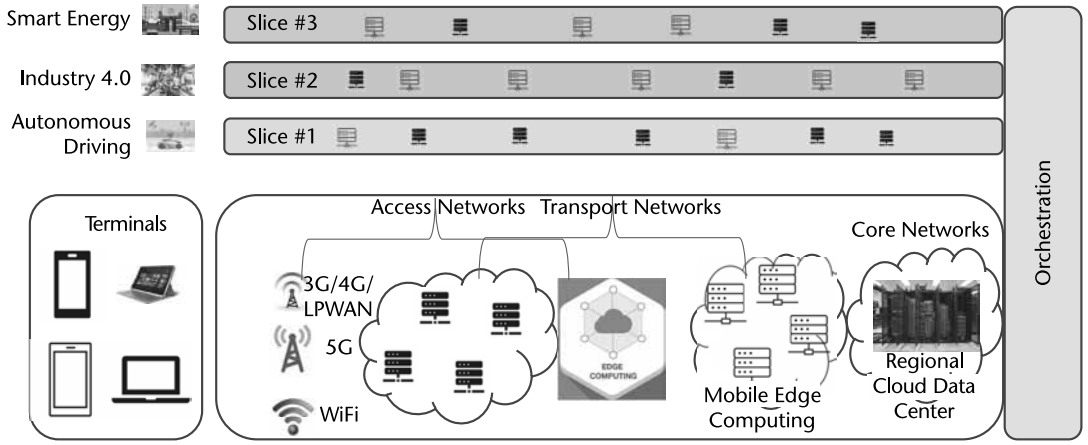

Figure 9.1 End-to-end connectivity for IoT applications in a 5G environment.

nationwide coverage. The 5G deployment roadmap is largely driven by the timelines defined by 3GPP with respect to the 5G New Radio (5G NR) standard. The latter includes several milestone releases, which have gradually been completed and deployed during the last few years.

In line with the 3GPP roadmap, field trials have been taking place since 2018. Existing 5G deployments are focused on supporting use cases like faster download of video on smartphones and mixed reality (i.e., combined augmented and virtual reality) over wide area networks (WAN). Moreover, private indoor 5G networks for residential and enterprise applications are nowadays feasible. In the coming few years, 5G stakeholders will focus on the commercialization of the technology (i.e., its business use beyond field trials and interoperability tests).

9.2 Blockchain Technology

9.2.1 Introducing Bitcoin and Blockchain Technology

In recent years, there has been a surge of interest on cryptocurrencies like Bitcoin and Ethereum, which are based on distributed ledger infrastructures and blockchain technology. The Bitcoin cryptocurrency was the first large-scale, decentralized payment system based on blockchain technology. It was introduced in 2008 in the scope of a famous paper by Satoshi Nakamoto [5]. Over the years, it became a blockbuster currency, and it is currently traded and exchanged with major paper currencies like the U.S. dollar and the Euro.

Bitcoin is a combination of several things:

- A currency;
- A payment system;
- A collection of algorithms and software implementations.

The algorithms and software implementations of Bitcoin make it a distributed, decentralized, digital currency system. In simple terms, it can be considered as a bank run by an ad hoc network. The revolutionary feature of the Bitcoin system is that it can operate without a need for a conventional trusted third party, like conventional banks. Rather, it leverages a distributed transaction system that allows all participants to a Bitcoin network to ensure the validity of payment and money transfer transactions by means of appropriate digital checks. In this way, Bitcoin enables payments with low transaction costs. It can also provide anonymity, as the transmission of a valid and validated customer identify is not yet a prerequisite for performing transactions.

Technically, all participants of the Bitcoin network maintain full transaction history in a ledger, as illustrated in Figure 9.2. Participants can perform a transaction (e.g., sending money to another participant) after referencing previous transactions in the ledger. This referencing is required in order to ensure that the sender possesses the money to be paid. In the scope of a Bitcoin payment, money is sent to the public key (i.e., the address) of the recipient. A digital signature is needed to unlock and spend funds.

Overall, Bitcoin's blockchain technology is based on hashing and digital signatures. Hashing provides a way for everyone on the blockchain to agree on the state of the blockchain. Digital signatures provide a way to ensure that all transactions are only made by the rightful owners. Likewise, hashing and digital signatures ensure that the blockchain has not been corrupted or compromised. One of the main value propositions of blockchain technology is its strong security and robustness, but also the fact that it cannot be easily compromised, as it does not have a single point of failure.

Bitcoin has also introduced the powerful, yet computationally expensive concept of mining. It refers to the process of validating Bitcoin transactions based on

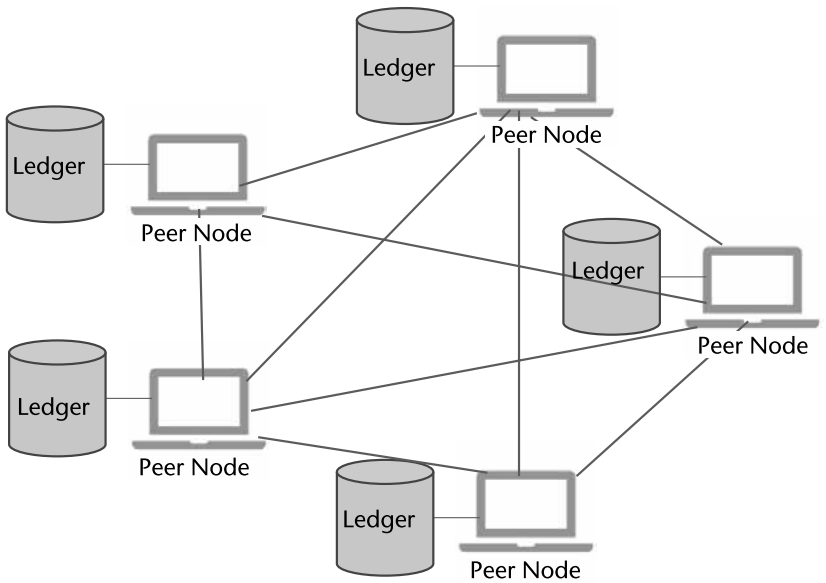

Figure 9.2 High-level overview of a blockchain network.

solutions to complex hashing problems. Specifically, miners (i.e., transaction validators) perform difficult computations (i.e., called "proof of work") in order to secure transactions into the chain of blocks. Miners are participants to the Bitcoin network, which are rewarded with Bitcoins for successfully mining transactions. This is also the way that Bitcoins are created. In the earliest days of Bitcoin, mining was done with CPUs from normal desktop computers. Nowadays, the Bitcoin network has grown larger and the mining process requires significant computational resources. Hence, graphics cards, or graphics processing units (GPUs), are more effective at mining than CPUs and have gradually become dominant in the Bitcoin mining processes. Eventually, the application-specific integrated circuit (ASIC) has been designed to support Bitcoin mining. The first ASICs for Bitcoin were released in 2013 and have been improved upon since, with more efficient designs coming to market. Mining is competitive and today can only be done profitably with the latest ASICs. When using CPUs, GPUs, or even the older ASICs, the cost of energy consumption can be greater than the revenue generated from the mining process.

The Bitcoin network and cryptocurrency is quite unrelated to IoT systems and applications. Nevertheless, its underlying blockchain technology is likely to be used in conjunction with IoT platforms and devices as discussed next.

9.2.2 Ethereum and Smart Contracts

From a computational and functional perspective, the Bitcoin blockchain provides the means for auditing the validity of monetary transactions. However, it does not support the execution of more sophisticated programs (i.e., of more complicated pieces of business logic). Over the years, blockchains that enable execution of entire programs, which are called smart contracts [6], have emerged. The first and more prominent such blockchain is the Ethereum blockchain.

The main building block of Ethereum is smart contracts, which provide executable Turing complete programs over the distributed blockchain infrastructure. A smart contract is a computer program that lives inside the Ethereum network and has its own memory, code and monetary balance (i.e., "Ether" balance). Every time a participant of the blockchain sends a transaction to the smart contract, the application logic of the smart contract is executed. The latter logic entails storage of data, conducting transactions and interacting with other contracts. Smart contracts are maintained by the Ethereum network, without central ownership or control. They can be programmed in languages familiar to any programmer, yet the most popular language for writing Ethereum smart contracts is Solidity.

Based on the smart contracts' capability, Ethereum is a more general blockchain infrastructure than Bitcoin, which is a currency-oriented infrastructure only. Each Ethereum node has a virtual machine forming a planetary-scale computer. Virtual machines run smart contracts, while users can call functions on the contract (i.e., they can carry out transactions).

Smart contracts live on the Ethereum blockchain and have their own Ethereum address. They can send and receive transactions. Whenever they receive a transaction, they are activated. They can also be deactivated when the transaction is complete. Moreover, they are associated with a fee-per-CPU usage and storage

step. Overall, they can be thought as a novel decentralized programming construct that comes with the benefits of a blockchain (i.e., security, lack of a single point of failure, and transparency). As such, they enable a novel programming paradigm that is characterized by decentralization. Indeed, smart contracts are run in the different nodes without being subject to some central control and orchestration like conventional computer programs. Hence, they can enable new forms of decentralized applications that can be extremely handy for solving some types of business problems. However, Ethereum smart contracts also have limitations when compared to state-of-the-art, large-scale programming environments. First, the Ethereum virtual machine is slow and hence not suitable for large computations. Second, storage on the blockchain can become expensive. Third, smart contracts cannot effectively deal with very large datasets (e.g., big data applications). Finally, in a smart contract's infrastructure, the decentralization versus scalability trade-off should be considered. These limitations have given rise to the development of private blockchain infrastructures that provide increased scalability at lower costs, while at the same time offering fine-grained control over who can participate in the blockchain network.

9.2.3 Uses of Blockchain Technology

Blockchain technology is what makes the above-listed cryptocurrencies (i.e., Bitcoin, Ethereum) work. Its main building block is a distributed ledger, which is used for sharing information instead of a central database. The distributed ledger is made up of blocks of data that are chained together. Cryptography, hashing, and digital signatures make it almost impossible to make changes once something is recorded. This ensures the integrity and anti-tampering properties of the blockchain. Hence, a blockchain comprises immutable, time-stamped records, which guarantee transparent recording and documentation of the information shared among the participants.

Blockchain enables a new paradigm of decentralized applications, notably applications that can be run without central control. Such applications provide value in very specific cases, where the use of a central database is not a good or viable alternative. Specifically, blockchain technology could be considered in cases of distributed applications where all the following conditions apply:

- There is a need for sharing and exchanging data through some database.
- Multiple writers who do not trust each other must access the shared database.
- There are reasons to avoid a trusted third party as intermediary.
- There is a need for interactions between different transactions in the shared database.

When at least one of the above conditions does not apply, a central database can provide an effective alternative. For example, in cases where there is a trusted third party for the problem at hand, a central database could be deployed instead of a blockchain infrastructure. In recent years, there has been hype around the use

of blockchain technology. The above criteria can provide a useful guide for dispelling this hype and deciding when blockchain could really be a good fit.

Assuming that the above conditions are met, blockchain infrastructures provide the means for writing and executing smart contracts (i.e., computer algorithms that can automatically execute the terms of a contract). Typical examples of blockchain smart contracts can be found in:

- *Supply chain management applications:* For instance, one company can pay a specific amount (e.g., $4,500) to another company upon the shipment of an order. A smart contract can be written to conduct and synchronize the transactions of delivering the order, sending a proof of delivery from the recipient to the sender and concluding the payment. These transactions can be executed on a decentralized fashion (i.e., without the need for a third party that will monitor and validate the delivery and the payment).
- *Multisided marketplaces:* Smart contracts could be used to provide decentralized trust between the participants of any multisided marketplace (e.g., a taxi app marketplace, a residency rental marketplace, a crowd-funding network). Instead of relying on a third party that holds a commission fee for its services, participants could leverage smart contracts that execute and validate the terms of their business relationship.
- *Industrial IoT applications:* Consider, for example, a renewable energy sources network of prosumers (i.e., actors being both consumers and producers of energy). A smart contract can be used to negotiate the transfer of energy from one party to another, in cases where one participant needs power and the other has excess power available.

9.2.4 Blockchain Technology and Internet of Things

9.2.4.1 Motivation

In recent years, there have been many research works that suggest the use of blockchain technology as an enabling infrastructure for IoT applications. The rationale behind these propositions is mainly twofold:

- *To alleviate the scalability limitations of the future IoT applications:* Existing IoT architectures scale quite well for large IoT deployments that comprise many thousands of objects. However, in the future, it is predicted that deployments with many millions or even billions of IoT devices will emerge. Hence, future cloud infrastructures will have to handle trillions of transactions between IoT devices, which could question their scalability. In this context, decentralized models could nicely complement existing edge or cloud infrastructures towards enhancing their scalability.
- *To support actuation capabilities that will not be subject to centralized control:* The proliferation of smart objects and CPS as part of IoT deployments will increase the number of applications that will employ actuation towards

influencing the physical world. This proliferation of actuation decisions will hardly be supported by a centralized paradigm that will control all objects' interactions. Instead, mechanisms for decentralized control over actuation mechanisms should be sought. Blockchain could be one such mechanism.

9.2.4.2 Decentralized IoT Applications

Overall, blockchain technologies could enable IoT objects and entire IoT ecosystems to interact in a decentralized and more scalable fashion, beyond the traditional broker-based networking paradigm. For example, devices could be authenticated in the blockchain rather than on a central cloud server or databases Such authentication mechanisms will leverage multiple nodes and could be much more scalable than state of the art cloud mechanism. Likewise, blockchain technologies will support the formation of secure mesh networks of IoT devices.

The concept is illustrated in Figure 9.3, which shows how a conventional edge or cloud IoT application for supply chain management can be implemented based on distributed ledger technology (i.e., blockchain). The edge or cloud IoT application collects supply chain information (e.g., objects track and trace information) based on multiple IoT devices that interact with edge servers. The information ends up in a centralized cloud database, where centralized application logic (e.g., objects' traceability) can be implemented. However, the blockchain applications implement no centralized application logic. Rather, traceability information is kept by all the nodes of the blockchain network in their ledger. Likewise, the logic of the traceability application is completely decentralized and implemented based on smart contracts that are executed in the various peer nodes. A smart contract defines the conditions under which the condition of a physical object should be changed in the

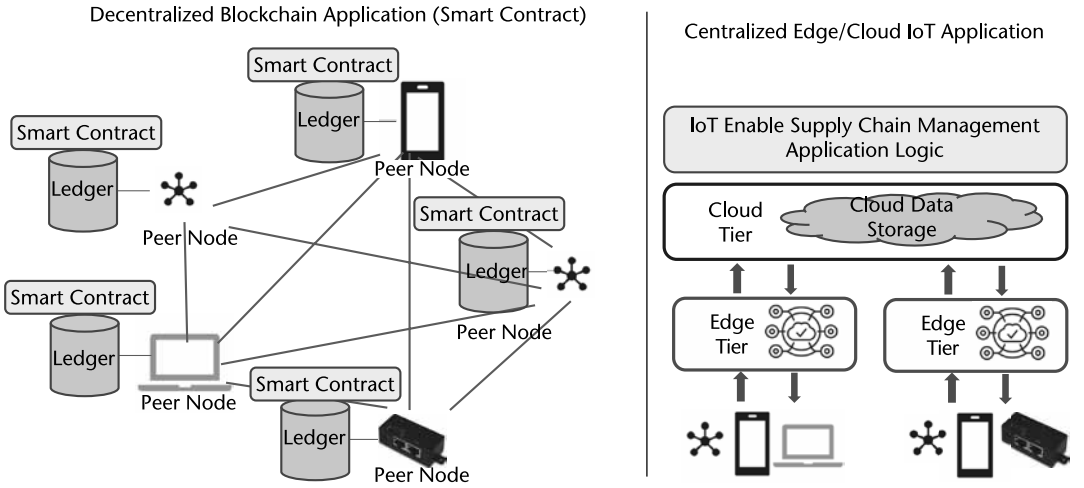

Figure 9.3 Decentralized blockchain-based IoT application (left) versus conventional centralized edge or cloud IoT application (right).

ledgers of the various nodes. For instance, the status of an order is not maintained in centralized cloud storage, but rather kept in the distributed ledger of the peer nodes. Likewise, state changes in the order are performed in the ledger of the peer needs in line with their smart contract. The latter may require certain conditions to be met. In the case of the decentralized application, all peer nodes agree on a single version of the truth about the supply chain, which is recorded in their ledgers. This is a radically different approach from the conventional edge or cloud paradigm, where a central entity is in charge of determining and updating the status of the supply chain, following the collection and aggregation of information from all the IoT devices of the supply chain deployment.

In principle, the blockchain-based IoT paradigm could be used to replace centralized edge or cloud IoT applications. Nevertheless, the two paradigms can be also combined in ways that take advantage of both the centralized and decentralized world. This is illustrated in Figure 9.4, which presents the formation of a decentralized network of objects that belong to different edge or cloud infrastructures (i.e., to different providers or operators). These objects could collaborate autonomously in terms of their mesh network towards providing intelligence at a point of interest, without a need for central coordination by a cloud operator.

Within such mesh networks, device communications and information exchange will be reliable and not subject to vulnerabilities such as device spoofing. The blockchain's potential for IoT security was presented in Chapter 7.

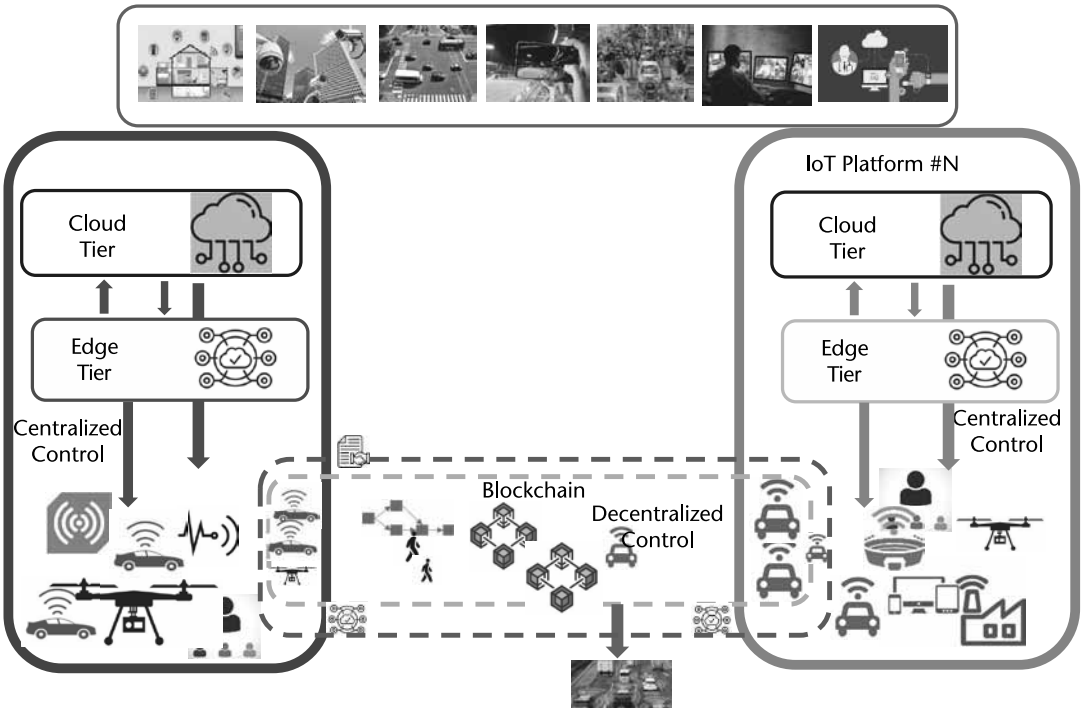

Figure 9.4 Decentralizing the edge or cloud paradigm for IoT applications using blockchain technology.

9.2.4.3 Blockchain IoT Platforms

Several blockchain-based architectures for IoT have been recently introduced as a means of decentralizing the convention edge or cloud paradigm for IoT applications. Indicative examples include:

- Modum.io, which exploits the ledger in order to ensure that sensitive goods remain in proper conditions throughout their life cycle;
- The Blockchainofthings.com, which has established a blockchain infrastructure in order to ease the implementation of secure IIoT applications.

Furthermore, there are several open-source projects that enable enterprises to develop their own blockchain infrastructures as a means of managing digital assets and securely exchanging data (e.g., Openchain and Hyperledger Fabric, both from the Linux Foundation). Several research works have also introduced blockchain-based systems, leveraging on the decentralization, transparency, auditability, and security capabilities of blockchain technology. A list of such platforms is available in recent surveys about IoT and blockchain (e.g., [7–9]). Finally, several proof-of-concept implementations for IoT applications have been carried out in the scope of research projects (e.g., [10]).

9.3 AI Trends for IoT

9.3.1 Overview

The future of IoT applications goes together with the evolution of AI. Machine learning is already widely used for mining IoT data, as presented in Chapter 6. In the medium term, AI technologies such as robotics and machine learning will be increasingly deployed as part of IoT systems. Therefore, AI trends will have an impact on the design, development, deployment, and operation of IoT systems. Here, three of these trends are presented:

- Explainable AI (XAI), which will be aimed to increase the transparency and trustworthiness of IoT systems;
- AI security technologies, which will be destined to protect IoT systems from cyber-security attacks against their AI modules;
- Automated machine learning (AutoML) technologies, which will boost the use of AI and IoT systems by nonexperts.

9.3.2 XAI

Many AI systems (e.g., deep learning systems) operate as "black boxes" that do not provide insights on the details of their operations. This results in a lack of

transparency about their operation and acts as a setback to their acceptance by humans. Human users need to understand the operation of AI systems in order to trust them. In this context, there has recently been a surge of interest in the interpretability and explainability of AI systems, which is largely motivated by the need to ensure the transparency and accountability of AI operations, as well as by the need to minimize the cost and consequences of poor decisions. This interest is also in line with mandates about ensuring the ethical nature of AI systems. In Europe, for example, the High Level Expert Group (HLEG) of the European Union has released some guidelines for ethical AI. The transparency and explainability of AI operations have a prominent position among these guidelines.

XAI research aims at providing a set of techniques that produce more explainable models (e.g., machine learning and deep learning models), while maintaining a high level of searching, learning, planning, and reasoning performance. XAI enables and facilitates human users to understand, trust, and effectively manage the emerging generation of AI systems. Recently, many research works have introduced different measures and frameworks for XAI. Most of these frameworks focus on defining model explainability, formulating explainability tasks for understanding model behavior, developing solutions for these tasks, and specifying techniques for evaluating the performance of models in explainability tasks. The spectrum of proposed methods is very wide as it includes methods for all the different types of AI systems.

Interpretability of deep neural networks is often algorithm-specific and has few human-subject tests to verify whether proposed methods indeed enhance human interpretability [11]. One popular class of techniques is coined as perceptive interpretability. The latter include the interpretabilities that are rather obvious and can be humanly perceived (e.g., algorithms that classify some objects in given segments). Perceptive interpretability methods include the saliency techniques, which explain the decision of an algorithm by assigning values (e.g., probabilities or heatmaps) that reflect the importance of input components in their contribution to that decision [12, 13].

Feature extraction can also provide insights on explainability and interpretability of AI models through identifying the features that are the strongest predictors of the AI outcomes (e.g., [14]). Another class of techniques leverages game theory to interpret the predictions of machine learning and deep learning models (e.g., [15]).

The vast majority of XAI applications are currently found in the medical-healthcare domain [11], as well as in finance and transport sectors. However, there are a lot for techniques for explainable robots (i.e., methods that attempt to explain the behavior of robots [16]).

In the medium term, IoT systems will incorporate AI techniques in order to provide insights in the operation of their AI parts. In the future, you should not be surprised in case you will be given the possibility to ask your autonomous car why it is driving in a certain way. The autonomous car will leverage XAI techniques to provide insights on its driving behavior, thus increasing the trust of the humans on it.

9.3.3 AI Security for IoT Systems

The expanded deployment of AI systems as part of IoT deployments raises new cybersecurity challenges (i.e., challenges relating to attacks against the AI part of the system). In order to leverage the potential of AI, future IoT systems will have to confront these challenges. Next we introduce two AI-related attacks that must be confronted in the scope of the development and deployment of next generation IoT systems.

9.3.3.1 Evasion Attacks

Evasion is one of the most common run-time attacks against deep learning systems. Such attacks present the deep neural network with adversarial inputs that cannot be correctly identified by it [17]. For instance, in an autonomous driving context, an adversary could present the self-driving car with a differentiated stop sign that cannot be correctly classified by the deep neural network used by the car. This could have a disastrous impact on the operation of the vehicle.

In the scope of evasion attacks, attackers seek to create such modified, adversarial input (i.e., adversarial examples) that looks like the original input.

The research community is already working on techniques that can successfully confront evasion attacks. The latter include:

- *Formal methods for testing and verification of the AI/IoT system:* Formal methods provide rigorous and well-structured mathematical representations of the operation of deep neural networks. As such, they can serve as a guide for testing them exhaustively with many different inputs. Such testing could proactively unveil malicious input patterns in order to identify and apply remedial actions accordingly.
- *Training the systems with adversarial examples (i.e., adversarial learning):* Another solution is to train the deep neural network with adversarial examples. In this way, it will be able to operate correctly even in the presence of adversarial input. Nevertheless, adversarial training must be preceded with a process that helps to identify adversarial examples.

9.3.3.2 Poisoning Attacks

Poisoning attacks take place at the training time of the machine learning systems (e.g., of the deep neural network). It entails the contamination of the data that are used to train a classifier towards compromising its operation [18]. Data contamination can take place in different ways such as through modifying labels, through injecting malicious data points in the training data, and through contaminating the algorithm used by the classifier. Poisoning attacks are expected to become increasingly important in IoT systems, given that the latter will be frequently retrained as new datasets become available. Frequent retraining will provide more opportunities for poisoning attacks.

The first step to confronting a data contamination attack is to identify it. In this direction, XAI techniques can be used to explain its operation. The outcomes of XAI techniques can alert AI and IoT system operators about possible abnormal behaviors following the training or (re)training of the machine learning or deep learning systems. Accordingly, techniques for identifying and removing adversarial training data can be applied.

9.3.4 Automated Machine Learning and AI

Nowadays, the development and execution of AI systems (such as machine learning systems) hinges on the involvement of AI experts. In the future, it is important to develop AI systems for the nontechnical users. Likewise, it is critical to facilitate the process of developing, (re)training, and deploying AI systems for real-world problems. One of the AI trends in this direction involves automating the process of applying machine learning to real-life applications. In this direction, a wave of AI-based techniques for the automated development and execution of data pipelines is developed and promoted under the term automated machine learning (AutoML) [19]. AutoML is largely about automating the machine learning process, as a means of boosting the engagement of nonexperts. Along with increased automation, AutoML can boost the development of simpler, yet powerful solutions (i.e., solutions that outperform conventional machine learning). To this end, AutoML tools that can automatically test and evaluate alternative machine learning models are developed. Such tools can automatically identify the structure of the data, apply relevant data preprocessing functions, and select the most appropriate models that yield high performance.

9.4 Summary

The IoT paradigm is advancing in a rapid pace. In addition to innovations at the level of smart objects and internet-connected devices, IoT advances are driven by a rich set of emerging technologies. This chapter has focused on three of these technologies, namely, 5G, blockchain, and AI. This chapter has shed light on how these technologies operate and what is their relation to IoT systems and applications. It has also provided a vision about how these technologies will advance IoT in the years to come. The aim of this chapter was not to provide an exhaustive presentation of these emerging technologies, but rather to indicate some of the most important innovations that will shape the next generation of IoT applications. IoT developers and deployers must watch out for these technologies in the years to come.

Chapter 10 is devoted to the presentation of a set of popular IoT applications. These applications integrate and use many of the technologies that were presented in this chapter and in previous chapters.

References

[1] Solomitckii, D., et al., "Technologies for Efficient Amateur Drone Detection in 5G Millimeter-Wave Cellular Infrastructure," *IEEE Communications Magazine*, Vol. 56, No. 1, January 2018, pp. 43–50.

[2] Akpakwu, G. A., et al., "A Survey on 5G Networks for the Internet of Things: Communication Technologies and Challenges," *IEEE Access*, Vol. 6, 2017, pp. 3619–3647.

[3] Chettri, L., and R. Bera, "A Comprehensive Survey on Internet of Things (IoT) Toward 5G Wireless Systems," *IEEE Internet of Things Journal*, Vol. 7, No. 1, January 2020, pp. 16–32.

[4] Miladinovic, I., and S. Schefer-Wenzl, "NFV Enabled IoT Architecture for an Operating Room Environment," *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, Singapore, 2018, pp. 98–102.

[5] Nakamoto, S., "Bitcoin: A Peer-to-Peer Electronic Cash System," Cryptography Mailing list, 2009, https://metzdowd.com.

[6] di Angelo, M., and G. Salzer, "A Survey of Tools for Analyzing Ethereum Smart Contracts," *2019 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPCON)*, Newark, CA, 2019, pp. 69–78.

[7] Atlam, H. F., et al., "Blockchain with Internet of Things: Benefits, Challenges, and Future Directions," *Int. J. Intell. Syst. Appl.*, 2018.

[8] Panarello, A., et al., "Blockchain and IoT Integration: A Systematic Survey," *Sensors*, Vol. 18, 2018, p. 2575.

[9] Fernandez-Carames, T. M., and P. Fraga-Lamas, "A Review on the Use of Blockchain for the Internet of Things," *IEEE Access*, 2018.

[10] Isaja, M., and J. Soldatos, "Distributed Ledger Technology for Decentralization of Manufacturing Processes," *ICPS*, 2018, pp. 696–701

[11] Došilovic, F. K., M. Brcic, and N. Hlupic, "Explainable Artificial Intelligence: A Survey," *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, Opatija, Croatia, May 21–25, 2018.

[12] Jacovi, A., O. S. Shalom, and Y. Goldberg, "Understanding Convolutional Neural Networks for Text Classification," *Proc. of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, Brussels, Belgium, 2018.

[13] Tulio Ribeiro, M., S. Singh, and C. Guestrin, "'Why Should I Trust You?' Explaining the Predictions of Any Classifier," *Proc. of 22 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August 2016, pp. 1135–1144.

[14] Ma, W., et al., "Probabilistic Representation and Inverse Design of Metamaterials Based on a Deep Generative Model with Semi-Supervised Learning Strategy," *Advanced Materials*, Vol. 31, No. 35, 2019.

[15] Lundberg, S. M., and S. -I. Lee, "A Unified Approach to Interpreting Model Predictions," *NIPS*, 2017, pp. 4768–4777.

[16] Sheh, R. K. -M., "Why Did You Do That? Explainable Intelligent Robots," *AAAI Workshops*, 2017.

[17] Kwon, H., H. Yoon, and D. Choi, "Priority Adversarial Example in Evasion Attack on Multiple Deep Neural Networks," *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, Okinawa, Japan, 2019, pp. 399–404.

[18] Xiao, H., et al., "Support Vector Machines Under Adversarial Label Contamination," *Neurocomputing*, Vol. 160, July 2015.

[19] Cai, H., et al., "AutoML for Architecting Efficient and Specialized Neural Networks," *IEEE Micro*, Vol. 40, No. 1, January-February 2020, pp. 75–82.

CHAPTER 10

IoT Applications

Following a presentation of the main IoT technologies in previous chapters, this chapter discusses how these technologies are enabling novel IoT applications in a variety of sectors. This chapter presents some of the most prominent application areas for IoT, with an outlook towards emerging applications that combine conventional IoT devices, smart objects, cloud computing, and AI. In coming years, a scale-up of the presented applications will be enabled by emerging IoT technologies such as 5G and cloud or edge computing.

The application areas addressed in this chapter span the areas of smart cities, wearables, healthcare, and connected transport. Emphasis is also paid in the presentation of IIoT use cases, which have a growing momentum given their proven return on investment (ROI).

10.1 Smart Cities

10.1.1 Smart City Definitions

For over a decade several enterprises, public organizations and research firms have provided definitions about the notion of a smart city. Here are the smart city definitions from three major research firms:

- Forrester Research has defined a smart city: "A city that uses information and communications technologies to make the critical infrastructure components and services of a city—administration, education, healthcare, public safety, real estate, transportation, and utilities—more aware, interactive, and efficient" (https://www.urenio.org/2010/12/04/forrester-research-on-smart-cities/).
- Gartner's definition of smart city is: "A smart city is based on intelligent exchanges of information that flow between its many different subsystems. This flow of information is analyzed and translated into citizen and commercial services. The city will act on this information flow to make its wider ecosystem more resource-efficient and sustainable. The information exchange is based on a smart governance operating framework designed for cities sustainable" (https://www.gartner.com/en/documents/1615214).
- IDC Energy Insights in 2011 defined smart cities: "'Smart city' refers to a local entity—a district, city, region or small country—which takes a holistic approach to employing information technologies with real-time analysis that

encourages sustainable economic development" (https://www.globalsecuritymag.com/IDC-Energy-Insights-Launches-the,20120411,29559.html).

In line with these definitions, the term smart city refers to a novel vision for urban development. The smart city development vision is characterized by investments in human capital and technology infrastructures towards enabling sustainable development, economic growth, and improved QoL for the citizens. In this direction, one of the key characteristics of a smart city is its ability to manage resources (e.g., natural resources such as energy and water) in efficient ways.

IoT is the most prominent technology behind smart cities [1]. As illustrated in Figure 10.1, a variety of IoT devices and services can be integrated in the smart city environment to enable a host of novel applications such as traffic management, energy management, urban security, and ambient assisted living. The latter can be implemented based on the state-of-the-art edge or cloud paradigm for architecting and deploying IoT systems.

10.1.2 Smart City Drivers

The smart city concept is motivated by the need to solve real problems in an era of increased urbanization [2]. Since ancient times, people have been gathering in cities and urban environments, as this facilitates collaboration and economies of scale. In recent years, urbanization trends have been exploding, which is one of the major factors that has led city authorities and citizens to seek new efficiencies in urban development. In this context, the introduction of the smart city concept is driven by four factors:

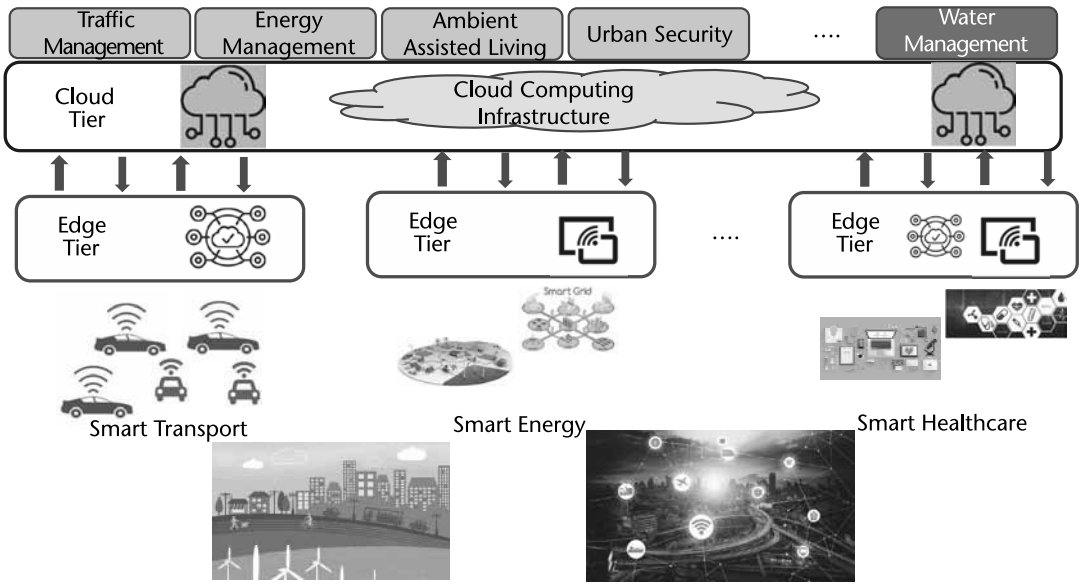

Figure 10.1 Overview of a smart city environment.

- *Urbanization trends:* The urban population worldwide is currently over 3.5 billion people, while growing at a rapid pace. Specifically, it is expected that the urban population will double by 2050. This growth leads to resource depletion pressures and asks for efficient ways for managing city resources. Furthermore, urbanization trends intensify phenomena such as exclusion, inequality, and rising insecurity challenges for significant parts of the cities' population.
- *Demographic changes:* We are currently witnessing ongoing changes in the demographics of modern cities. For example, the number of seniors who were 60 years of age or over is the fastest-growing segment of the population. Furthermore, there is a decline in infant mortality. Along with high fertility rates, this decline leads to a proliferation of the younger population. Such demographic changes ask for policies that can provide increased employment opportunities for younger individuals, while supporting the elderly in aging well.
- *Changing lifestyles:* Life in modern cities is influenced by changes in family patterns, as well as new habits in work and mobility. Prominent examples of such habits include teleworking, vehicle sharing, and shared renting of houses. Hence, there is a need for novel urban services that can provide support for these changes.
- *Climate change:* Cities are also influenced by climate changes and the global warning phenomenon. Thus, there is also a need for policies and infrastructures that facilitate efficient use of water, energy, and other resources, along with other measures for sustainable growth.

Table 10.1 exemplifies how IoT applications provide solutions to these challenges.

10.1.3 IoT for Smart Cities

IoT technologies are essential for the successful implementation of the smart city urban development vision. Smart cities are empowered by the internet-based connectivity of sensors and devices that are deployed in the urban environment. Market-wise smart cities are expected to be one of the settings where the highest monetary value for IoT will be generated. According to a 2015 McKinsey report

Table 10.1 IoT Use Cases Alleviate the Challenges of Urban Environments

Challenge	*Solution*	*Related IoT Use Cases*
Urbanization trends leading to resource depletion	Optimize usage of resources such as energy and water	IoT-enabled smart grid; smart water management
Demographic changes lead to aging population	Facilitate the elderly to live longer independently	IoT-based, ambient-assisted living
Changing lifestyles leading to new mobility patterns	Provide platforms for sharing bicycles and vehicles	IoT-enabled, connected vehicles facilitating pooling

[3], IoT will generate up to $11.1 trillion a year in economic value by 2025 and smart cities will be one of the IoT settings with the highest business value.

As already presented in Chapter 1, IoT is not about a single technology but rather about a pool of related technologies that are combined and integrated to empower IoT applications [4, 5]. This is one of the main advantages of IoT for smart city deployments, as it enables IoT applications to meet the diverse requirements of smart city needs. Several of the IoT technologies that were presented in previous chapters are very relevant to smart cities and are widely deployed in smart city settings. For instance, Wi-Fi, 4G/LTE, and 5G technologies are used for connectivity across urban computing infrastructures and devices. As another example, sensor and IoT middleware platforms are used to provide the digital plumbing for the interconnection of different IoT devices in the city. Likewise, cloud computing is increasingly used for the scalable processing of data from the many thousands of devices that are typically deployed in the urban environment. Finally, IoT analytics, including data processing, data mining, machine learning, and big data technologies, are used to enable data-driven decisions, actuation functionalities, and urban planning activities. The latter are based on the scalable processing of large amounts of IoT data streams from the smart city infrastructure.

10.1.4 IoT Standards and Vertical Applications for Smart Cities

Large-scale IoT deployments in smart cities make extensive use of IoT standards [5]. Depending on the target application in the smart city context, three different categories of standards are used:

- *IoT (connectivity) standards:* Examples are ZigBee, 3GPP, LoRa, oneM2M, and Wi-Fi, which provide the means for interconnecting devices and platforms in scalable ways. Connectivity standards serve the IoT infrastructure and are typically applicable to all the different vertical applications of the smart city.
- *IoT standards for vertical applications:* These are focused on providing support for specific vertical applications. For example, there are standards dedicated to smart home applications (e.g., UPnP, KNX[1]), standards pertaining to manufacturing and industrial applications (e.g., Industrial Internet Consortium (IIC) standards), and standards dedicated to urban transport and mobility (e.g., Car2Car Communication Consortium standards and standards of the Open Automotive Alliance (OAA)).
- *Standards exclusively developed to support smart city applications:* For example, the ISO 37120:2014 ("Indicators for City Services and Quality of Life") specifies indicators for QoL in a smart cities context, as well as standards dedicated to IoT applications integration and interoperability standards (e.g., Hypercat.io). Recently, several SDOs (e.g., IEC, ISO, ITU, IEEE, CEN-CENELEC, and ETSI) have announced joint efforts about smart

1. KNX is an open standard for commercial and domestic building automation.

city interoperability standardization towards the interworking of existing standards.

Smart cities represent a vibrant and growing market. This market is usually segmented and monitored based on the different vertical applications that address different sectors. Prominent examples of vertical applications in smart cities include:

- *Smart buildings:* These use IoT technologies in order to automatically control the building's operations. Specifically, smart buildings control and optimize processes associated with heating, ventilation, air conditioning, lighting, and security.
- *Smart energy management:* This aims at optimizing the production, distribution, and use of energy resources in the city. Such applications include the management and use of renewable energy sources.
- *Smart transportation and urban mobility:* These aim at optimizing the effectiveness, sustainability, and cost-efficiency of urban transport. They involve the management and balancing of multiple transport modalities including public transport, on-demand services, vehicle sharing, bike sharing, and ride hailing.
- *Smart urban security:* This employs IoT technologies in order to increase the security and safety of the city, through handling both cybersecurity and physical security aspects. Urban security applications are used to confront security incidents and combat terrorists.
- *Smart healthcare:* This aims at providing IoT-enabled, high-quality healthcare services to citizens in ways that reduce the pressure to healthcare systems. Smart healthcare applications empower health professionals in the city to access and analyze more data about the patients and the healthcare systems towards optimizing medical decisions. For example, in the scope of the recent novel coronavirus (COVID-19) pandemic, several cities have deployed platforms that enable them to access and analyze relevant datasets towards improving their COVID-19 outbreak-related decisions.

10.1.5 Maturity Model for IoT in Smart Cities

The development of IoT infrastructures and applications in a smart city context is usually a gradual and lengthy process. In most cases, the process is driven by an urban development plan and an accompanying IoT systems development plan. Hence, smart city infrastructures and applications are usually developed based on a phased approach, which is driven by some maturity model. A maturity model describes the evolution of the development of a smart city. It is used to assess the implementation progress of a smart city development plan. To this end, maturity models are usually composed of different development phases. A simple maturity model for smart city development comprises the following phases:

- *Phase 1—Digital Infrastructure Establishment:* During this phase, cities design and deploy infrastructural projects, such as the establishment of broadband networking and sensor networking infrastructures. These infrastructures transform a city to a digital city (i.e., they advance their digital maturity). The availability of digital infrastructures is a prerequisite for the transformation of a city to a smart city. As part of this phase, the various infrastructures are certified and validated in terms of their proper operation. Extensions and enhancements to a city's digital infrastructure can take place in subsequent phases as well.
- *Phase 2—Services Development:* In this phase, cities develop IoT applications and services over the established digital infrastructures. Most of the services concern vertical application areas, such as smart energy, smart transport, and urban mobility. In the scope of the development of these applications, all stakeholders become engaged in the smart city vision. Hence, both human capital and digital infrastructures are exploited.
- *Phase 3—Services Integration and Citizens Participation:* In this phase, services are integrated, and citizens' participation is enhanced. The integration of services enables a holistic approach to achieving goals set in the city's urban development plan. This holistic approach emphasizes the accomplishment of city-wide development goals (e.g., a target for the reduction of CO2 emissions or a target associated with increased use of public transport) based on multiple IoT projects. For example, many different projects contribute to a city's environmental performance and sustainability. The integration of these projects facilitates their collective exploitation towards achieving city-wide key performance indicator (KPI) about environmental performance. The key aspects on this integration are:
  - The integration of functionalities from multiple services to enable entirely new functionalities;
  - The ability to repurpose and reuse IoT data and services across different vertical applications areas;
  - The expanded participation of citizens across the entire life cycle of smart city services including service design, development deployment, and operation.

National Institute of Standards and Technology (NIST) has provided a framework for smart grid (i.e., smart energy) applications, which can help the understanding of the different layers and building blocks of a smart city deployment. The framework covers technical, information processing, and organizational factors.

From a technical perspective, the framework specifies the needs for:

- *Basic connectivity:* This comprises mechanisms for the physical and logical interconnection between the various smart city systems.
- *Networking interoperability:* This provides the means for interconnecting infrastructures that are based on different networking and IoT technologies, such as different wireless networks and optical networks.

- *Syntactic interoperability:* This enables the different smart city systems to understand the structure of messages exchanged between them.

From an informational viewpoint, the NIST framework specifies the following layers:

- *Semantic interoperability:* This ensures that diverse IoT systems understand the concepts entailed in the exchanged messages. At this layer, different smart projects and applications perceive smart city concepts (e.g., sustainability, CO2 emissions, transport routers, geographic locations) in the same way.
- *Business context:* This is the layer exploiting the unified understanding of concepts in order to model business knowledge and interactions in a unified way. At this layer, the exchanged information is put in the right business context including specific locations, business processes, times, and reasoning about how and why a specific event occurs.

The layers that deal with organizational factors implement high-level processes that are typically part of a city's urban development plan. They cover:

- *Business procedures:* This is the layer that implements business procedures of the city by leveraging information and services exchanged by the various systems.
- *Business objectives:* This is a layer that synthesizes various procedures to ensure that certain business objectives of the city are met.
- *Economic and regulatory policies:* This is the layer that ensures the implementation of the city's policies, including compliance to regulations (e.g., environmental or economic regulations). This layer leverages services of the business objectives layer, given that any policy can be expressed as a collection of business objectives.

The NIST framework leverages technological infrastructures and technical concepts to implement business objectives and policies.

10.1.6 IoT Applications' Interoperability in Smart Cities

For over a decade, cities have planned, developed, and deployed many smart city projects. In most cases, cities have deployed multiple systems, which have been deployed independently from each other, forming disaggregated silos (i.e., "smart islands") that do not interact with each other [6]. This leads to data fragmentation and the lack of easy ways to exploit multiple projects and datasets towards a city's business objectives. For example, in many cities, there are multiple projects that collect data for sustainability and environmental performance purposes such as smart transport, smart energy, and smart water management projects. The fragmentation of these datasets makes it challenging to combine them for urban planning pur-

poses. This raises the issue of data and semantic interoperability across different smart city applications [7].

Integrated smart city platforms facilitate the interoperability between different systems and applications. Interoperability can accordingly ease the integration of interrelated applications in the city, towards integrated optimizations that meet city-wide development goals.

10.1.7 IoT in Smart Cities and Open Data

Open datasets are among the key elements of a city's smart infrastructure. Open data can be combined with data from IoT devices to empower data-driven decision-making and optimization of city processes. In this context, the use of open datasets enables:

- Open and transparent governance of the city, as data about the city's decisions are openly available;
- Increased engagement of citizens, who are provided with open access to smart city data;
- Increased innovation as datasets facilitate the attraction and engagement of innovators in the development of novel smart city projects.

Smart cities are increasingly implementing open data portals, where they publish open datasets. One of the most prominent examples of open data portals is the London datastore. The datastore enables citizens, business owners, researchers, and developers with access to over 700 datasets that help them to understand the city and develop solutions to London's problems.

10.1.8 Trends in IoT for Smart Cities

Three of the most prominent smart city development trends nowadays concern interoperability, citizen engagement, and public private partnerships. Specifically:

- *Interoperability [6]:* This refers to the need for ensuring semantic and business interoperability across independently developed applications. As already outlined, this is a prerequisite to implementing integrated smart city strategies.
- *Citizen engagement:* This is a key ingredient of every smart city strategy. There is a growing trend towards engaging citizens and other smart city stakeholders across all stages of the IoT services life cycle, including design, development, deployment, and operation. As part of this trend, IoT services development is increasingly carried out based on cocreation approaches, which emphasize stakeholders' collaboration for IoT services design during specially structured cocreation workshops.
- *Public private partnerships:* These are important to financing smart city infrastructures and projects. Financing of smart city infrastructures is a very

challenging task, given that infrastructure investments are very costly and feature low ROI and quite long payback periods. In several cases, digital infrastructures are developed and used to support projects of high societal value, which do not necessarily have a positive ROI. For these reasons, IoT and broadband infrastructures are usually financed with the involvement of both the public and the private sectors. In this direction, there is a rise of public-private partnerships for the establishment of city infrastructures. A prominent example of such a public-private partnership is the LinkNYC initiative in New York. LinkNYC is an infrastructure project that created a network offering free Wi-Fi service across the city. The project has been expanded to other cities as well.

The longer-term viability and sustainability of IoT deployments in smart cities hinges on the introduction and validation of proper business models that can monetize IoT services. Thus, many cities are investigating and validating such models [8].

10.2 Wearables

10.2.1 Introducing Wearables

Wearable computing enables a class of very interesting IoT applications, given that many internet-connected devices are wearable. Wearables are small electronic devices that comprise one or more sensors. They are typically associated with clothing or worn accessories, such as watches, wristbands, glasses, and jewelry. Wearables come with some sort of computational capability that enables them to capture and process data about the physical world. Many wearables are also equipped with a display of some kind, which enables them to visualize data.

Wearable devices are not always directly connected to the internet. In most cases, they offer connectivity capabilities such as Bluetooth or near-field communications (NFC) that enable them to connect to smartphones. The latter serve as their gateways that enable wearables to connect to the internet.

Wearable devices come with a wider range of inputs and output devices. Input devices include keyboard alternatives (including chording keyboards and special-purpose keyboards), mouse alternatives (including trackballs and joysticks), tab alternatives (including buttons and dials), eye trackers, head trackers, pens, gesturing, barcode readers, textiles, video capture devices, microphones, GPS locators, speech recognition modules, and skin sensors. Likewise, output devices examples include head-mounted displays (HMDs), flat panels, text-to-speech (TTS), tactile output, nonspeech auditory output, and paper.

Wearables' functionalities are empowered by their sensors. Depending on the type of wearable devices, several sensors can be attached to it, including light sensors, sound sensors, acceleration sensors, and humidity sensors. These sensors enable wearables to be used in the scope of a wide range of applications, including both consumer-oriented ones and nonconsumer-oriented ones (e.g., industrial). Typical examples of consumer-oriented applications can be found in the areas of

fitness and sports, fashion and apparel, home automation, retail, infotainment, and gaming. However, nonconsumer-oriented applications are found in the areas of defense and security, manufacturing, healthcare, and more.

10.2.2 IoT Wearables

As outlined in earlier chapters, IoT's value stems from the integration of data and services from multiple devices. The IoT paradigm adds value to wearables, thanks to its ability to combine their data and services, but also to integrate them with the capabilities of other (wearable and nonwearable) internet-connected devices. IoT wearables add information and value to wearables' capabilities, through additional sensors and devices. The term "IoT wearables" is therefore introduced to differentiate them from single wearables devices that are not combined with other IoT devices.

The evolution of smartphones and wearable connectivity technologies (e.g., Bluetooth) is among the catalysts of the wearables IoT paradigm. Different types of wearable devices have been around for decades, but without smartphones their ability to connect and to interact with other devices was very limited. Figure 10.2 illustrates the concept of IoT wearables. It presents how different wearable devices can be interconnected through state-of-the-art smartphones to other parts of the IoT ecosystem in order to support services in areas such as healthcare, fitness, and employees' safety.

10.2.3 Examples of IoT Wearable Devices and Ecosystems

Some examples of popular wearable devices that are increasingly used in IoT applications are:

- *Apple Watch:* A very popular smartwatch, which includes a heart rate sensor, GPS, and an accelerometer. It is fully integrated into the Apple ecosystem and interacts flexibly with other Apple devices such as the iPhone and the iPad.
- *Sensoria Fitness T-shirt:* A T-shirt that comprises embedded textile sensors and enables tracking of the heart rate.

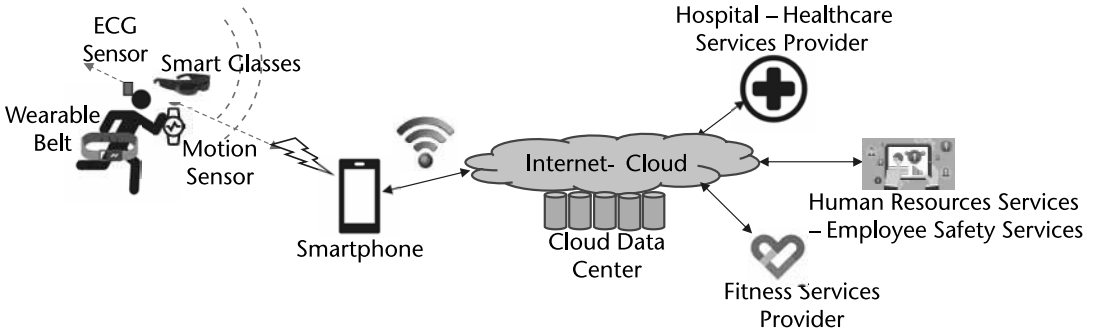

Figure 10.2 IoT wearables environment.

- *Adidas Smart Run:* A wrist device that monitors the wearer's heart rate and location data. It is fully blended into Adidas miCoach system, which provides several apps for athletes and fitness professionals.
- *FitBit's Flex:* A sleek wristband that provides real-time statistics on a user's daily fitness activity.
- *Nike+ Sportwatch:* A watch that measures the distance traveled by athletes. It also measures the pace and speed of the wearer's run.
- *Samsung's Galaxy Gear:* It is an Android-based smart watch. It synchronizes with a cellphone in order to provide smartphone-like capabilities.
- *Garmin SmartWatch:* A watch that comes with built-in sports apps. It provides wireless connectivity and enables a more active lifestyle for the wearer through a range of motivational features.

10.2.4 IoT Wearable Trends

The future of wearable devices will be based on with their integration into the IoT paradigm. This integration has already started given that wearable devices provide services as part of wider ecosystems that provide complete programming and application development environments beyond the device level. These ecosystems enable the development of applications based on the data and functionalities of the device in the cloud. This is the first step for the integration of wearables in the IoT ecosystem (i.e., in cloud IoT platforms provided by major vendors, including wearables' devices vendors).

The IoT wearables market has a positive momentum. According to market research in December 2019 from Meticulous Market Research Pvt. Ltd., the global wearable devices market is expected to grow at a compound annual growth rate (CAGR) of 11.3% in the period 2019 to 2025 to reach $62.82 billion by 2025. This growth is propelled by the growing consumer preference for connected devices and the rapid increase in the number of IoT devices. The main barrier that need to be overcome to realize the growth potential of wearables relates to data privacy and security, as well as to the lack of sufficient connectivity infrastructures in developing and underdeveloped countries.

Wearables ecosystems are currently created around the devices of a single vendor (i.e., they are vendor-specific). Hence, they do not offer interoperability between devices and applications of different vendors. In the coming years, we expect to see interoperability across devices of different types and from different vendors, but also across different wearables ecosystems. This could empower users to manage their personal data across all wearables' ecosystems. Furthermore, the integration of wearable devices in the wider IoT ecosystem will enable the development, deployment, and operation of integrated wearable IoT services combining data and services from multiple devices and ecosystems. The development of such novel wearable services will be driven by business requirements in sectors where wearables add value, including fitness, healthcare, and industry.

10.3 IoT in Smart Manufacturing: Industrial IoT (IIoT)

10.3.1 Introduction to IIoT and Industry 4.0

IoT is currently enabling the next generation of production and manufacturing systems. As discussed in Chapter 1, future manufacturing is shaped by the advent of the fourth industrial revolution (Industry 4.0) and related trends. In this context, the main drivers of future manufacturing include [9]:

- *Shift from capacity to capability:* Future plants are expected to be very flexible and hyper-efficient. They should enable manufacturers respond to variable market demand and achieve high levels of customer fulfillment.
- *New production models:* The future of manufacturing will enable more customized and demand-driven production models, beyond conventional mass production. Specifically, future factories will take advantage of digital technologies such as IoT, cloud computing, and big data to implement a transition from make-to-stock (MTS) production models to make-to-order (MTO), configure-to-order (CTO), and engineer-to-order (ETO) production models.
- *Profitable proximity sourcing and production:* Future factories will produce modular products based on common platforms and configurable options. In this direction, manufacturers will be gradually adopting hybrid production and sourcing strategies. As part of these strategies, modular platforms will be produced centrally, while at the same time enabling suppliers, distributors, or retailers to tailor final products locally to better serve local customer demand.
- *Improved workforce engagement:* Despite increased automation, people will remain at the center of the factories of the futures. Specifically, workers will provide the degree of flexibility and decision-making capabilities required to deal with increasing operational complexity. In this context, future factories will also enable higher levels of collaboration between workers, but also between workers and machines.

IoT technologies enable the digitalization of industrial processes and facilitate the implementation of new production models. They also facilitate production flexibility and effective workforce engagement. Industry 4.0 is largely based on the introduction of CPS within industrial plants, as well as with their use to digitize industrial processes. Hence, the term industrial IoT (IIoT) has been introduced to denote the IoT technologies that are deployed in industrial applications and empower Industry 4.0. In essence, IIoT refers to the extension and the deployment of IoT systems in industrial sectors such as manufacturing, supply chain management, energy, oil and gas, and mining.

In recent years, we have witnessed a rapid digital transformation of industrial organizations, based on CPS systems and IIoT technologies. The latter facilitate the digitization of physical processes and enable the collection of large amounts

of digital data about them. Based on the analysis of these datasets, industrial organizations derive unique insights on production processes. They can also provide actionable recommendations for optimizing them. Furthermore, based on CPS and IoT systems, it is possible to close the loop to the field, which opens up new horizons in industrial automation.

CPS systems are among the catalysts of the emergence and the rise of IIoT. Industry 4.0 and IIoT are also empowered by the accelerated evolution of digital technologies such as cloud computing, big data, AI, cybersecurity, and augmented reality (AR). These technologies provide the means for collecting, persisting, analyzing, processing, and visualizing very large data volumes in secure and effective ways. As such, they are considered as the digital pillars of Industry 4.0 [10].

Although Industry 4.0 is in its early stages, there are already applications and use cases that have a proven and tangible ROI. These are driving the adoption and growth of IIoT in industrial enterprises in a variety of industrial sectors, including not only manufacturing, but also energy, oil and gas, mining, and logistics. These use cases are also driving the rapid growth of relevant market segments. According to a recent study by Grand View Research, Inc., the smart manufacturing market is estimated to reach $514.3 billion in U.S. dollars by 2027, registering a CAGR of 11.8% over the period 2020 to 2027.

10.3.2 Popular IIoT Use Cases

10.3.2.1 Flexible Production Lines

CPS and IoT systems are configurable and programmable, which enables the reconfiguration of production processes. This is the foundation for flexible production lines that can rapidly embrace changes in the configuration of production operations and of the devices used. For example, flexible production lines can be reconfigured to take advantage of additive manufacturing processes (i.e., the use of a 3-D printer) for producing materials or parts of a product. The reconfiguration can be realized based on digital functions and at quite short time scales, rather than based on tedious changes to the configuration of the operation technology (OT) of the shopfloor.

Most importantly, flexibly reconfigurable production lines can enable the production of customized products as part of the shift towards mass-customization. Specifically, CPS and IoT systems ease the implementation of changes to the configuration of automation devices, which enables the customization of production operations. In this way, Industry 4.0 is driving a transition from conventional mass-production models to customized production.

10.3.2.2 Condition Monitoring and Predictive Maintenance

Enterprise maintenance processes are usually carried out based on a preventive approach. The latter entails service, repair, or replacement of assets in specified time intervals before the equipment breaks down. Hence, in a preventive maintenance approach service times are calculated empirically, considering the average life

expectancy of the asset. The latter is usually indicated by the original equipment manufacturer (OEM). Preventive maintenance prevents unscheduled downtime and production stoppage, yet it is far from being optimal in terms equipment productivity and overall equipment efficiency (OEE).

Industry 4.0 enables a shift from preventive maintenance to condition-based maintenance (CBM) of assets and equipment [11]. Specifically, CPS and IoT devices enable the collection of very large volumes of data about the condition of the assets. These data stem from a variety of different sources including vibration sensors, thermal images, acoustic sensors, ultrasonic sensors, temperature measurements, power consumption metrics, oil analysis measurements, and business information systems (e.g., Enterprise Resource Planning (EPR) and Quality Monitoring and Management (QMM) systems). These datasets are processed using big data analytics and AI algorithms to predict when an asset will need maintenance and to decide the best point in time to schedule the maintenance activities. For example, using predictive analytics on data about the condition of an asset, it is possible to derive credible estimates about its remaining useful life and end of life. This is a key to optimizing maintenance schedules. Likewise, prescriptive analytics can be used to provide actionable maintenance insights that optimize OEE. The latter optimization is grounded on the fact that accurate remaining useful life estimations improve the usage and productivity of the asset, when compared to empirical assessments of the asset's lifetime.

The use of IIoT technologies for extracting predictive insights in enterprise maintenance activities is called predictive maintenance. Predictive maintenance employs CBM towards deriving timely predictions about the health status of the assets. It is considered a "killer" Industry 4.0 use case due to its very broad applicability. Specifically, predictive maintenance is applicable to all industrial sectors where industrial assets and equipment are used, including manufacturing, oil and gas, mining, construction, and smart buildings [12].

10.3.2.3 Digital Quality Management (DQM) and Zero Defect Manufacturing (ZDM)

The digital transformation of the shopfloor is opening new horizons in the quality management of industrial processes. By collecting and analyzing digital data about industrial assets and processes, it is possible to predict defects ahead of time. Moreover, it is possible to derive insights about the root causes of quality issues. Accordingly, CPS and IoT systems can be employed to alleviate and remedy defects. Remedial strategies can be implemented not only reactively (i.e., after the occurrence of the defect), but also proactively (i.e., before the defects happen). This digitization of quality management processes boosts their accuracy and eases the effective implementation of quality management disciplines such as SixSigma and Total Quality Management (TQM). Moreover, it increases the levels of automation and facilitates the extraction of knowledge about potential problems.

In the coming years, Industry 4.0 will enable a shift to digital quality management (DQM), which shall provide a digital implementation all processes of the ISO 9000 quality management system. When applied to manufacturing, a DQM

approach can also enable the vision of zero defect manufacturing (ZDM) [13]. The latter optimizes multiple aspects of production processes at the same time, such as quality, cost, time, and energy efficiency. In this context, IIoT technologies enable the implementation of complex analytical models about production optimization towards the ZDM vision.

10.3.2.4 Supply Chain Management and Optimization

Nowadays, supply chain optimizations are based on the management of digital information that is provided by the various participants to the chain. IIoT systems enrich this information with data about the physical processes, thanks to the deployment and use of CPS in the shopfloor. In this way, IIoT enables the vision of the connected and informed plant.

Connected and informed plants access detailed information about production and logistics operations across the entire value chain. Hence, they can develop real-time knowledge about events and processes that affect their operation, such as knowledge about the arrival of spare parts for maintenance, information about delays in the shipment of source materials, real-time access to customer orders, and more. IIoT-enabled supply chains offer richer information about distributed virtualized manufacturing processes, which enables new efficiencies and optimization opportunities.

10.3.2.5 Scalable Digital Simulations

Digital simulation is a very useful tool for industrial organizations as it enables them to execute what-if scenarios and to evaluate the efficiency of alternative automation and control workflows. IIoT technologies boost the scalability and accuracy of simulation processes, thanks to the employment of big data technologies for the collection and persistence of large volumes of process data. This broadens the scope of the processes that can be simulated, while enhancing the credibility of the simulation results.

10.3.2.6 Digital Twins

Digital Twin is a revolutionary concept that goes together with IIoT. A Digital Twin is a faithful representation of production systems and processes in the digital world [14]. It includes information about physical assets, processes, people, physical locations, systems, and devices of an industrial organization or a supply chain. As the state of these entities change, so does their representation in the digital twin. To this end, the development of a digital twin blends AI, big data, and IoT technologies to ensure that the twin keeps up with the status of the physical world. Moreover, digital twins integrate historical data in order to incorporate knowledge about the past status and behavior of physical entities. Historical data can be then used to predict and anticipate future behaviors of the twin.

Digital Twins enable a wide range of use cases. For instance, they can create realistic and accurate what-if simulations of physical processes, considering

real-world data, in addition to simulated datasets. The objective of these simulations is to identify optimal production configurations prior to their actual deployment in the shopfloor. As another example, Digital Twins can facilitate accurate predictions about the future state of assets, notably predictions that leverage detailed information about the surrounding environment of the asset.

Digital Twins are currently designed, implemented, and deployed for a variety of production processes in various industrial sectors. As such, they are one of the most characteristic applications of the Industry 4.0 era.

10.3.2.7 Workers' Training and Safety

Industry 4.0 is set to eliminate laborious and error-prone processes. The latter are gradually taken over by industrial robots and other automation systems. Despite the proliferation of such automation systems, humans remain at the heart of the production processes, as they are still the most intelligent and flexible production resource. In this context, IIoT empowers use cases destined to improve the human centricity and the safety of production processes.

IIoT technologies are used to facilitate workers' training. Technologies such as augmented reality (AR) and virtual reality (VR) enable the creation of interactive cyber-representations of the production environment, as a means of training workers on how to perform production tasks. The respective training environments are safe, given that they obviate the need for placing workers in hazardous environments. Likewise, they can be used to deliver training in remote locations. Training on how to properly perform digitally enabled production tasks is a key to preparing workers for the Industry 4.0 era, through proper upskilling and reskilling processes.

IIoT technologies can be also used to boost workers' safety in human-in-the-loop production tasks (i.e., tasks involving both humans and automation systems). As a prominent example, human-centered Digital Twins can be used to monitor workers' characteristics such as performance and fatigue during certain tasks. In this way, tasks that could jeopardize the workers' safety can be identified. Furthermore, CPS and smart objects such as drones and automated guided vehicles can be used to replace humans in hazardous environments. For instance, unmanned aerial vehicles (UAVs) are increasingly used for inspections of assets under harsh conditions. Finally, AR technologies are commonly used to configure and control automation devices and smart objects from remote. This obviates the need for manufacturing workers to act in hazardous environments, while saving them the extra effort of moving or traveling to actual location of the task.

10.4 IoT in Healthcare

10.4.1 Overview

There is a close affiliation between healthcare applications and IoT, given that a wide range of IoT devices are deployed and used in healthcare settings [15]. For example, a modern healthcare system comprises IoT devices such as the following:

- Sensors that are used to collect patient data;
- Microcontrollers that process, analyze, and wirelessly communicate the data;
- Microprocessors that enable rich graphical user interfaces;
- Healthcare-specific gateways enabling sensor data analysis, as well as transmission of healthcare data to a cloud infrastructure.

Hence, a typical healthcare deployment in a care center or even the home environment comprises multiple IoT devices, including wearables. These devices enable the acquisition of data about the patient. Patient data are collected or aggregated by some gateway and latter transmitted to the cloud in line with the cloud or edge computing paradigm. Through the cloud, patients' data can be accessed by many different users, such as physicians, practitioners, and other healthcare professionals.

According to a recent market research report published by Meticulous Research, the IoT in healthcare market is expected to grow at a CAGR of 29.9% from 2019 to 2020 to reach $322.2 billion by 2025. This growth is propelled by the constant pressure to reduce costs in the healthcare system, as well as by the proliferation of internet-connected devices and the evolution of high-speed networking technologies. However, the growth of IoT in healthcare is hindered by concerns associated with the security and protection of sensitive medical data.

10.4.2 Popular Healthcare Use Cases

10.4.2.1 Diagnostic Applications

The processing of data from IoT devices enables novel diagnostic applications [16]. The latter combine patients' lifestyle information derived from sensors (e.g., activity data) with vital signals from medical devices (e.g., cardio signals) and other diagnostic information (e.g., data from Laboratory Information Systems (LIS)) to provide more accurate and personalized diagnosis for specific diseases. In this direction, IoT-enabled diagnostic applications collect, integrate, and analyze information from multiple devices, including medical devices. The analysis may employ machine learning algorithms, such as those presented in Chapter 6.

There are also simpler IoT-enabled diagnostic applications that operate at the device level. Specifically, there are many IoT devices that are primarily used as diagnostic devices. Prominent examples include devices for vital signs data capturing, connected thermometers, sleep monitors, and biometric scanners.

10.4.2.2 Ambient-Assisted Living

The deployment and use of IoT devices in smart environment are also key enablers of the ambient-assisted living paradigm. AAL refers to the integration of ICT products and services in the social environment towards improving individuals' QoL. There are AAL applications that support individuals at all periods of their life. However, the majority of AAL applications target elderly or disabled individuals.

AAL applications make extensive use of IoT technologies to support end users in maintaining a healthy lifestyle, managing a disease, managing chronic conditions, and living longer independently. For example, IoT-enabled smart pillboxes notify patients and their caretakers about cases where a dose of the medicine seems to have been forgotten. There are also devices that support users in cases of falls (e.g., connected alarm buttons).

10.4.2.3 Clinical Trials

During the last couple of years, IoT has seemed promising to revolutionize clinical trials. Specifically, the use of IoT devices enables the collection of information about how a drug under trial affects specific patients' phenotypes. This can serve as a basis for more successful clinical trials that lead to more personalized and more effective drugs. For instance, it can facilitate the timely identification of adverse side effects on a patient during a clinical trial. The identification can be based on the monitoring of the vital signs of a patient.

As another example, personalized information about the effects of a drug on a patient can provide the investigator with unique insights about the efficacy of the drug on these specific patients and probably other patients with similar characteristics (e.g., age, daily habits, vital sign values). This could accordingly guide clinical trial investigators in understanding the types of patients for whom the drug exhibits the maximum efficiency.

Finally, IoT devices can greatly facilitate the patients' adherence to clinical trial protocols. For example, the IoT-enabled pillbox can provide tangible evidence that the clinical trial participant did take the drug as prescribed in the clinical trial protocol.

10.4.2.4 Clinical Care

Another healthcare-related area where IoT can add significant added value is the convergence of IoT with clinical care. This convergence aims at reducing the number of conventional (physical) visits of a healthcare professional to a patient or vice versa. To this end, the monitoring of vital signs and other examinations are performed remotely based on IoT devices and IoT technologies. This improves the patient's quality of care and lowers costs for the patient and the healthcare system. A typical IoT-enabled remote care scenario involves:

- Constant monitoring of the patient's state using IoT-driven, noninvasive sensors;
- Collecting comprehensive physiological information;
- Use of gateways and the cloud to analyze and store the required information;
- Sending the analyzed data wirelessly to caregivers for further analysis and review.

10.4.2.5 COVID-19 Applications

During the recent COVID-19 outbreak, several IoT applications have emerged to help citizens, businesses, and public organizations to cope with the effects of the pandemic. As a prominent example, connected thermometers are increasingly used at hospitals and at other public locations to screen citizens and staff based on the most common COVID-19 symptom (i.e., fever). Likewise, wearable devices such as bracelets and rings are used to constantly monitor patients' vital signs, including temperature, heart rate, and blood oxygen levels. The latter data are usually submitted to cloud platforms where they are processed in order to derive insights about the spreading of the disease and other epidemiological parameters.

There are also mobile applications that serve as symptom trackers and COVID-19 monitoring tools. These apps enable users to report their COVID-19 symptoms. Symptoms are accordingly submitted to healthcare organizations, municipalities, or other authorities, along with information about the vital signs and chronic conditions of the citizens.

Early in April 2020, Google and Apple announced their collaboration towards providing a contacts tracing infrastructure based on smartphones and the use of secure Bluetooth technology. This contacts tracing infrastructure is destined to be privacy-friendly, as it will not store the location and identity of the traced individuals. It will be provided to governments and healthcare organizations to facilitate contact-tracing procedures (i.e., the identification and notification of an infected person's contacts).

10.5 Connected Vehicles

10.5.1 Introduction

Intelligent transportation is one more area where IoT will be extensively deployed [17]. Connected vehicles provide some of the most prominent application IoT nowadays, while other important applications such as the self-driving car are emerging [18]. The rationale behind deploying IoT technologies in transport is that modern vehicles comprise tens of different sensors. These sensors are essentially connected to the internet, given that internet access is a standard feature of connected vehicles. Furthermore, they can flexibly connect with other devices inside the vehicle, which enables a wide range of interesting applications.

Connected vehicles enable many interesting and complex applications. They include numerous sensors and connected devices inside each car, while being able to interact with other connected vehicles and the smart transport infrastructure (e.g., Intelligent Transportation Systems). Popular edge or cloud architectures provide a good foundation for developing and deploying connected vehicles applications. In case of real-time decisions, edge computing functionalities are employed to provide low-latency functionalities close to the field. However, selected data from the vehicles can be also shared in the cloud to facilitate applications such as vehicle condition monitoring and traffic management.

As already outlined in Chapter 9, emerging connectivity technologies (like 5G) are essential for supporting connected transport applications. This is because the collection and real-time processing of data from the numerous connected devices of a vehicle require in-network data processing and very high data rates. Indeed, connected driving and autonomous driving are two of the applications that will be driving the deployment and evolution of 5G networks in the years to come. Overall, 5G networks will alleviate some of the major technical challenges of connected driving and autonomous driving, which are related to the very fast collection and processing of large volumes of data. Nevertheless, the challenges of connected driving and autonomous driving are not only technical. In the case of autonomous driving, there are also ethical challenges such as the need to properly balance the interests of the passengers with the safety of others (e.g., pedestrians or cyclists). The latter are expected to be addressed through proper legal and regulatory initiatives that will complement the ongoing technical developments.

10.5.2 Developing Connected Vehicle Applications

Several applications can be developed, deployed, and operated based on data from the sensors and the internet-connected devices [19]. Five different approaches can be used:

- *Running apps in the in-vehicle entertainment systems:* This is an approach supported by various toolkits such as Blackberry QNX CAR, Windows Embedded Automotive, Automotive Grade Linux, and Android.
- *Using a link to a smartphone that hosts and executes the apps:* This is a very popular approach, which is boosted by different vendors and initiatives such as Airbiquity, OpenCar, CloudCar, SmartDeviceLink, AppLink, MirrorLink, Apple CarPlay, Google Open Automotive Alliance, and Windows in the car.
- *Providing remote access to the vehicle through an API:* This approach is promoted by OnStar, General Motors API, Ford Remote API, and Airbiquity.
- *Access to data through the on-board diagnostics port called OBD-II:* This is the approach undertaken by Dash Labs, Mojio, Carvoyant, MetroMile, and smartdrive.io.

New and emerging initiatives are currently researched by W3C Automotive, the Web Platform Business Group, and OpenXC.

A prominent example of a smartphone-based approach to implementing and deploying applications for the connected car is provided by Apple's Car Play platform. The Car Play platform:

- Allows iPhone owners to use the features that they want in their cars without creating dangerous distractions;
- Enables pairing an iPhone with a vehicle. This is done by plugging it into the dashboard with a lightning cable. In such a case, the car automatically pops

up the CarPlay icon and updates compatible apps, while the phone screen is locked to eliminate any temptation to use it while driving.

Several automotive manufacturers are already supporting the Car Play platform.

10.5.3 Indicative Applications

Indicative applications for connected vehicles include:

- *Infotainment:* This brings information functions (i.e., navigation, location-based services, rear-seat web browsing, social networking) into the vehicle's entertainment system. For example, CarPlay can leverage iTunes to watch videos, run navigation apps on the in-dash display with a touch screen interface, and support Apple's voice companion Siri (vocal commands). In this way, it brings the entire Apple apps ecosystem to the dashboard and presents endless possibilities for an in-car experience. Example applications include reading out email and calendar reminders, ordering food, and switching on the heater.
- *Vehicle-to-vehicle (V2V) communication:* This enables wireless exchange of the position, speed, and location data between nearby vehicles. This enables a range of applications such as applications that improve the safety of commuters.
- *Vehicle-to-infrastructure (V2I) communication:* This enables wireless exchange of information between vehicles and roadside infrastructure. They enable the car to communicate with the road, digital signage, traffic lights, safety, and control systems. V2I applications can help avoiding crashes and traffic congestion, as they enable the connected car to have a very good understanding of the driving context both close to the car and in wider areas, based on information provided by Intelligent Transportation Systems.
- *Vehicles and smartphones:* These include two-way information exchange from the smartphone to the vehicle and vice versa. It also enables the exchange of On-Board Diagnostics (OBD/OBD-II) data, such as information regarding the engine and other crucial vehicle parameters with the smartphone. All these parameters can be displayed on the driver's smartphone and the same can be sent to the service provider for analysis. Other functionalities include the production of alerts (e.g., open doors, lights on, hand brake on) and the execution of actions such as locking and unlocking vehicle doors, rolling windows up or down, and tuning the air conditioning (AC) temperature.
- *Smartphone sensors for driving insights:* Smartphones have various sensors, such as accelerometer, gyroscope, or orientation sensor and GPS. By docking the smartphone to the vehicle, data from these sensors can be used to detect driving patterns, such as sharp turns, sudden acceleration, hard braking, drifting, and speeding. Furthermore, apps that profile the driver as safe or aggressive are developed in order to rate and compare different drivers and

to share such data with insurance providers. The latter can take advantage of these data to calculate premiums customized to the driver's behavior. The exploitation of smartphone sensors can also enable scenarios such as Pay-As-You-Drive (PAYD) and Pay-How-You-Drive (PHYD). The latter is among the upcoming offerings from automobile insurance companies that reward safe drivers and penalize rash ones with differential premiums.

- *On-board diagnostics for on-device analytics:* These are based on the OBD/OBD-II port that is commonly used in automobile service and maintenance. Faults, vehicle and engine speed, engine temperature, fluid levels, gear shifts, and battery status are all accessed regularly at vehicle repair shops. The ability to access them continually and derive up-to-date information enables post facto analysis. OBD information can be made available to the vehicle owners, giving them a better picture of the car's performance. Through monitoring these parameters actively and with some level of on-device analytics, drivers can get proactive service alerts on their smartphones and potential faults can be identified for early diagnosis and care. OBD-based applications are usually deployed as part of edge or cloud architectures.

10.6 Summary

This chapter focused on the practical integration and use of IoT technologies in real-life applications. The presented applications have tangible techno-economic value and high societal importance. They are already transforming public organizations and businesses alike. As illustrated in this chapter:

- Smart city applications have a significant impact on the citizens' quality of life, while contributing to the long-term sustainability of modern cities.
- IIoT applications are shaping the digitalization of the industry and providing new efficiencies for industrial enterprises.
- Wearables and healthcare applications are playing an important role in improving the quality of care and in alleviating the cost pressures on the healthcare systems worldwide.
- Connected transport opens new horizons in safe and comfortable driving.

This chapter provided a representative list of IoT-based applications. IoT software developers and innovators are nowadays offered with unprecedented opportunities for blending IoT data and services towards implementing novel concepts and applications. This has been made evident during the recent COVID-19 outbreak, where many innovative IoT systems were developed and used to help people fight against the pandemic. In the coming years, IoT technologies will play a significant role in propelling economic growth and in changing our lives for the better.

References

[1] Zanella, A., and L. Vangelista, "Internet of Things for Smart Cities," *IEEE Internet of Things Journal*, Vol. 1, No. 1, 2014.

[2] Shahidehpour, M., Z. Li, and M. Ganji, "Smart Cities for a Sustainable Urbanization: Illuminating the Need for Establishing Smart Urban Infrastructures," *IEEE Electrification Magazine*, Vol. 6, No. 2, June 2018, pp. 16–33.

[3] Manyika, J., et al., "Unlocking the Potential of the Internet of Things," McKinsey Global Institute, July 2015.

[4] Bellavista, P., et al., "Convergence of MANET and WSN in IoT Urban Scenarios," *IEEE Sens. J.*, Vol. 13, No. 10, October 2013, pp. 3558–3567.

[5] Kazmi, A. H., M. Serrano, and J. Soldatos, "VITAL-OS: An Open Source IoT Operating System for Smart Cities," *IEEE Communications Standards Magazine*, Vol. 2, No. 2, 2018, pp. 71–77.

[6] Schiele, G., J. Soldatos, and N. Mitton, "Moving Towards Interoperable Internet-of-Things Deployments in Smart Cities," *ERCIM News*, Vol. 2014, No. 98, 2014.

[7] Petrolo, R., et al., "Integrating Wireless Sensor Networks Within a City Cloud," *SECON Workshops*, 2014, pp. 24–27.

[8] Walravens, N., and P. Ballon, "Platform Business Models for Smart Cities: From Control and Value to Governance and Public Value," *IEEE Communications Magazine*, Vol. 51, No. 6, June 2013, pp. 72–79.

[9] Soldatos, J., et al., "Internet of Things Applications in Future Manufacturing," in O. Vermesan and P. Friess, (eds.), *Digitising the Industry Internet of Things Connecting the Physical, Digital and Virtual Worlds*, Denmark: River Publishers, 2016.

[10] Soldatos, J., O. Lazaro, and F. Cavadini, (eds.), *The Digital Shopfloor: Industrial Automation in the Industry 4.0*, Denmark: River Publishers, 2019.

[11] Hashemian, H. M., and W. C. Bean, "State-of-the-Art Predictive Maintenance Techniques," *IEEE Transactions on Instrumentation and Measurement*, Vol. 60, No. 10, 2011, pp. 3480–3492.

[12] Cachada A., et al., "Maintenance 4.0: Intelligent and Predictive Maintenance System Architecture," *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, Turin, 2018, pp. 139–146.

[13] Chiariotti, P., et al., "Smart Measurement Systems for Zero-Defect Manufacturing," *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, Porto, 2018, pp. 834–839.

[14] Barricelli, B. R., E. Casiraghi, and D. Fogli, "A Survey on Digital Twin: Definitions, Characteristics, Applications, and Design Implications," *IEEE Access*, Vol. 7, 2019, pp. 167653–167671.

[15] Zhu, H., et al., "Smart Healthcare in the Era of Internet-of-Things," *IEEE Consumer Electronics Magazine*, Vol. 8, No. 5, September 1, 2019, pp. 26–30.

[16] La, H. J., H. T. Jung, and S. D. Kim, "Extensible Disease Diagnosis Cloud Platform with Medical Sensors and IoT Devices," *2015 3rd International Conference on Future Internet of Things and Cloud*, Rome, 2015, pp. 371–378.

[17] Marosi, A. C., et al., "A Novel IoT Platform for the Era of Connected Cars," *2018 IEEE International Conference on Future IoT Technologies (Future IoT)*, Eger, 2018, pp. 1–11.

[18] Strobl, S., et al., "Connected Cars—Threats, Vulnerabilities and Their Impact," *1st IEEE Conference on Industrial Cyber-Physical Systems (ICPS 2018)*, Saint Petersburg, Russia, 2018, pp. 375–380.

[19] Wu, H., et al., "Developing Vehicular Data Cloud Services in the IoT Environment," *IEEE Transactions on Industrial Informatics*, Vol. 10, No. 2, May 2014, pp. 1587–1595.

About the Author

John Soldatos (http://gr.linkedin.com/in/johnsoldatos) received a Ph.D. in electrical and computer engineering from the National Technical University of Athens in 2000; has been an Honorary Research Fellow at the University of Glasgow, United Kingdom, since 2014; and is an adjunct lecturer at the University of Sheffield, United Kingdom (Athens Tech Campus). He was an associate professor and the head of the IoT Group at Athens Information Technology (AIT), Greece, from 2006 to 2019 and was an adjunct professor at Carnegie Mellon University, Pittsburgh, Pennsylvania, from 2007 to 2010. He has significant experience in working closely with large, multinational industrial enterprises (e.g., IBM, INTRACOM S.A, INTRASOFT International S.A, GFT Italia S.r.l) as a research and development consultant and delivery specialist, while being a scientific advisor to high-tech start-up enterprises offering IoT-based products and services, such as Innovation Sprint Sprl (Brussels, Belgium) and Innov-Acts Limited (Nicosia, Cyprus). Dr. Soldatos is an expert in IoT technologies and applications, including IoT applications in smart cities, finance (Finance 4.0), and industry (Industry 4.0). Dr. Soldatos has played a leading role in the successful delivery of more than 60 projects (commercial-industrial, research, and consulting), for both private and public-sector organizations, including complex integrated projects. In several of these projects, he had a leading role in the architecture and development of practical IoT systems. He is the cofounder of the open source platform OpenIoT (https://github.com/OpenIotOrg/openiot) and of the Edge4Industry (www.edge4industry.eu) community. He has published more than 180 articles in international journals and conference proceedings and chapters in books. He has also significant academic teaching experience, along with experience in executive education and corporate training in topics such as IoT, big data, artificial intelligence, and agile software engineering. Dr. Soldatos is a regular contributor in various international magazines and blogs on topics related to IoT, Industry 4.0, and cybersecurity. Moreover, he has received national and international recognition through appointments in standardization working group and expert groups and on various boards. Dr. Soldatos has led various working groups of the European Research Cluster on the Internet of Things (IERC), where he has been in charge of various white papers and technical reports. He has coedited and coauthored three edited books on IoT topics, including IoT for industrial automation (Industry 4.0), IoT analytics, and IoT security.

Index

D