CHAPTER

Internet-of-Things Systems

8

CHAPTER POINTS

- Internet-of-Things (IoT) = sensors + wireless networks + database.
- Wireless networking for IoT devices.
- Database design for IoT systems.
- Design example: smart home.

8.1 Introduction

The **Internet-of-Things** (IoT, or *Internet-of-Everything*) is a new name for a concept that has evolved over several decades. Mark Weiser of Xerox PARC coined the term *ubiquitous computing* to describe a range of smart and interconnected devices, as well as the applications to which these devices could be put. Moore's Law has allowed us to improve those early devices in several important ways: modern IoT devices provide more computing and storage; they operate at lower energy levels; and they are cheaper. Advances in wireless communications have also allowed these devices to communicate much more effectively with each other and with traditional communication systems. The result of these advances is an explosion in devices, systems, and applications.

This chapter describes the basic concepts underlying IoT system design. We start with a survey of applications that make use of IoT technologies in Section 8.2, followed by a discussion of IoT system architectures in Section 8.3. Section 8.4 studies several networking technologies, a central concept for IoT system design. In Section 8.5, we look at two types of data structures that can be used to organize information in IoT systems: databases and timewheels. We conclude with the example of an IoT smart home in Section 8.6.

8.2 IoT system applications

An Internet-of-Things system is a soft real-time networked embedded computing system. An IoT system always includes input devices: tags, sensors, and so on.

Computers as Components. https://doi.org/10.1016/B978-0-323-85128-2.00008-6

One may also include output devices: motor controllers, electronic controllers, and displays. The devices comprise data processing devices (displays, buttons) and cyber-physical devices (temperature sensors, cameras).

Application examples

IoT systems can be used for several different purposes:

- A computer-readable identification code for a physical object can allow a computer system to keep track of an inventory of items.
- A complex device (e.g., an appliance) can be controlled via a user interface on a cell phone or computer.
- A set of sensors can monitor activity with data analysis algorithms extracting useful information from the sensor data. Sports sensor systems, for example, can monitor and analyze the activity of an athlete or team of athletes. Smart building systems can monitor and adjust the temperature and air quality of a building.

Devices

We can identify three types of devices for people: **implanted** devices are within the body; **wearable** devices are worn outside the body; and **environmental** sensors are located entirely off the body (e.g., mounted on the wall). These categories also fit well with the objects. We use one set of terms for both cases: **interior**, **exterior**, and **environmental.**

Various interior or implanted devices are used for people, including heart rate sensors, neurological sensors, and so forth. For objects, sensors play an equivalent role: engine temperature sensors, and more. A smart band is an example of an exterior sensor for people. Environmental sensors range across many modalities, including door sensors, cameras, and weather sensors.

Example 8.1: Wall-Mountable Camera

An **Internet Protocol (IP) camera** is a video camera that transmits digital video over an Internet connection. Older video cameras transmitted analog video over cables. These cameras often transmit video in H.264 format; they may also use motion JPEG, a sequence of still frames. Many cameras also provide a still-image mode. A **pan-tilt-zoom** (PTZ) camera can move horizontally (pan) and vertically (tilt), as well as zoom its lens in and out. Some cameras use a semispherical reflector to capture panoramic images.

RFID

Radio-frequency identification (**RFID**) is an important class of IoT devices. An **RFID tag** can be used to provide an identification number for a physical object and possibly for other information as well. Many RFID tags are read-only; they can be programmed before installation using a separate machine. However, in use, they only reply to a request by transmitting their preprogrammed identification tag. Some tags can be written in use under the control of radio transmission.

We can identify two common use cases for RFID tag communication. The first is known as **passive** because the tag transmits only when it receives a request. An **active** tag will respond to requests but will also transmit on its own periodically.

We also use **passive** to describe RFID tags that have no internal power source; they receive all their energy from the outside. A battery-assistive passive device uses

passive communications but makes use of a battery. An RFID tag can use its antenna to receive radio-frequency energy. It can store some energy in a capacitor and use that energy to operate the radio to receive and transmit data.

RFID tags were designed to operate in several different frequency bands and at several different distance ranges. Some tags operate only within a few centimeters, whereas others can be read from dozens of meters away.

Some RFID tags use the electronic product code (EPC) [GS114] as an identifier. An EPC can be used to assign a unique name to a physical object.

8.3 IoT system architectures

Edge and cloud

We often divide an IoT system into **edge** and **cloud** components. An edge device is one of the system's application devices; data from the edge devices may be processed remotely at Internet servers referred to as "the cloud."

IoT use cases

We can identify several formulas for IoT design, each with its own associated use cases.

The simplest design formula is a smart appliance:

$$\text{IoT smart appliance} = \text{connected appliance} + \text{network} + \text{UI}$$

In this case, a communication node allows the appliance to connect to a user interface (UI) through the network, giving the user access to the controls and status of the appliance. Fig. 8.1 shows a UML sequence diagram for a smart appliance. In this scenario, a user interface runs on a device, such as a smart phone. The UI can interact

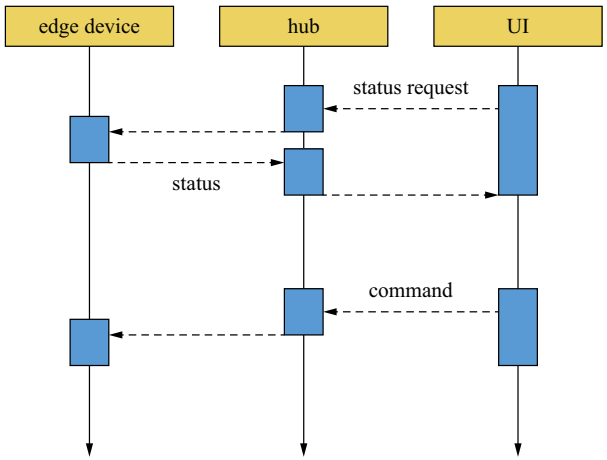


FIGURE 8.1

Use case for an IoT smart appliance.

with the smart appliance by sending and receiving messages via the hub. The UI can check the status of the smart appliance or give it commands.

A more sophisticated version applies to distributed sensor systems:

$$\text{IoT monitoring system} = \text{sensors} + \text{network} + \text{database} + \text{dashboard}$$

Examples of IoT monitoring systems include smart homes and buildings or connected cities. Fig. 8.2 shows a sample use case. The sensors feed data into a database via their hubs. A data analysis program runs in the cloud to extract useful information from sensor data streams. Those results are given to the user on a **dashboard** that provides a summary of the system's status, important events, and so on.

IoT networks can also be used for control, as illustrated in Fig. 8.3.

$$\text{IoT control system} = \text{sensors} + \text{network} + \text{database} + \text{controller} + \text{actuator}$$

A wireless sensor network can make sensor measurements that are sent to a cloud-based controller, which then sends a command to an actuator in the edge network. The control algorithm may be a traditional periodic algorithm, for example, a motor controller. It can also be an event-driven controller, as is typical for home or building energy management.

8.4 Networks for IoT

Networking is a critical component of an IoT system; wireless networking allows for a much wider range of sensor applications than is possible with wired networks. We

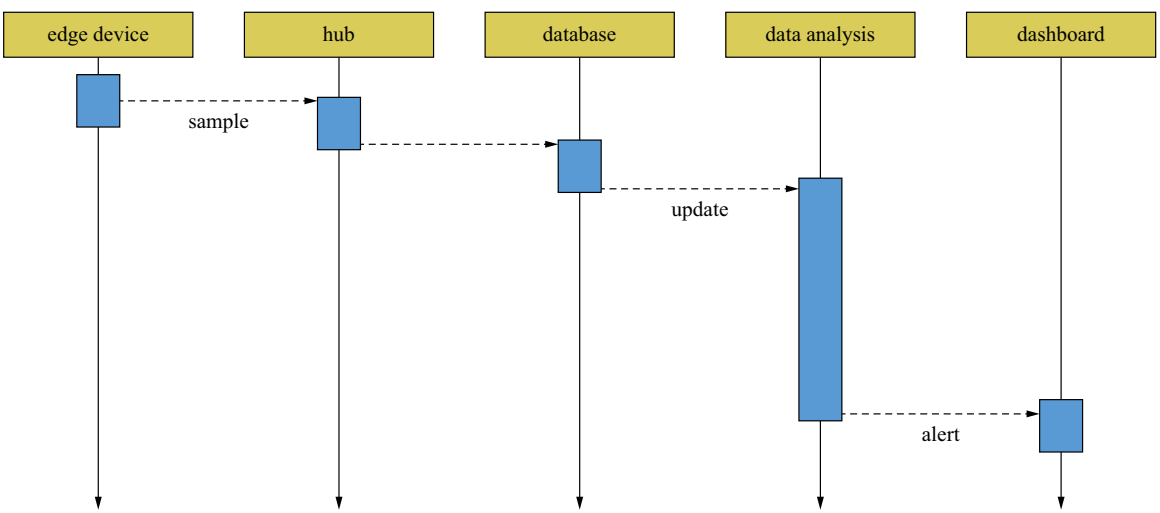


FIGURE 8.2

Use case for an IoT monitoring system.

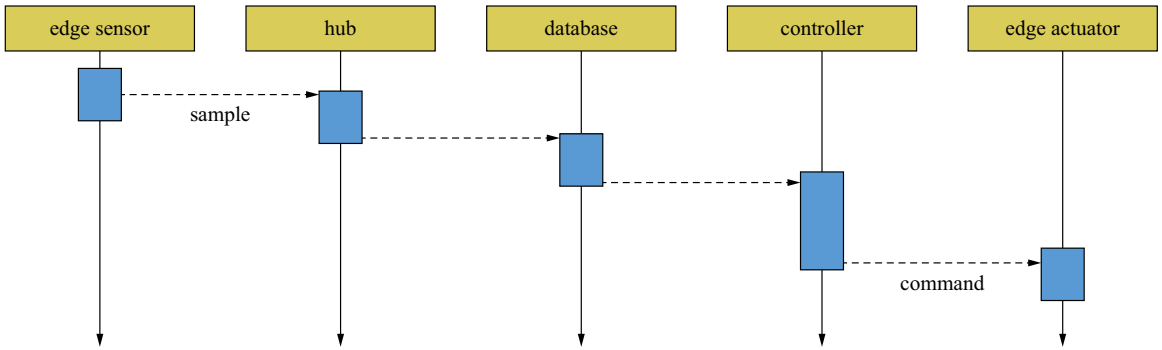


FIGURE 8.3

Use case for an IoT control system.

start with a review of the fundamental concepts in networking: the **Open Systems Interconnection (OSI)** model and the Internet Protocol. We then describe some basic concepts in wireless networks for IoT, followed by the specifics of four widely used wireless networks: Bluetooth/Bluetooth Low Energy (BLE), IEEE 802.15.4/ZigBee, Wi-Fi, and LoRa.

8.4.1 The open systems interconnection model

Networks are complex systems. Ideally, they provide high-level services while hiding many of the details of data transmission from the other components in the system. To help understand (and design) networks, the International Standards Organization has developed a seven-layer model for networks known as OSI [Sta97A]. Understanding the OSI layers will help us understand the details of real networks.

The seven layers of the OSI model, as shown in Fig. 8.4, are intended to cover a broad spectrum of networks and their uses. Some networks may not need the services of one or more layers. Higher layers may be missing, or an intermediate layer may not be necessary. However, any data network should fit into the OSI model.

OSI layers

The OSI model includes seven levels of abstraction, known as **layers**:

- **Physical (PHY):** The PHY layer defines the basic properties of the interface between systems, including the physical connections (plugs and wires), electrical properties, the basic functions of the electrical and physical components, and the basic procedures for exchanging bits.
- **Data link:** The primary purpose of this layer is error detection and control across a single link. However, if the network requires multiple hops over several data links, the data link layer does not define the mechanism for data integrity between hops but only within a single hop. The data link layer can be divided into two sublayers: **medium access control** (**MAC**) to control access permissions and the **logical link control** (**LLC**) layer, which encapsulates network protocols as well as managing error checking and frame synchronization.

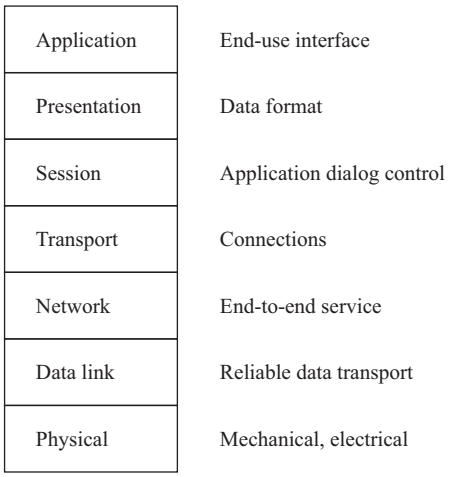


FIGURE 8.4

The open systems interconnection model layers.

- **Network (NWK):** This layer defines the basic end-to-end data transmission service. The network layer is particularly important in multihop networks.
- **Transport:** The transport layer defines connection-oriented services that ensure that data are delivered in the proper order and without errors across multiple links. This layer may also try to optimize network resource utilization.
- **Session:** A session provides mechanisms for controlling the interaction of end-user services across a network, such as data grouping and checkpointing.
- **Presentation:** This layer defines data exchange formats and provides transformation utilities to application programs.
- **Application:** The application layer provides the application interface between the network and the end-user programs.

Although it may seem that embedded systems would be too simple to require the use of the OSI model, the model is in fact quite useful. Even relatively simple embedded networks provide physical, data links, and network services. An increasing number of embedded systems provide Internet services that require implementing the full range of functions in the OSI model.

8.4.2 IP

The Internet Protocol (IP) [Los97, Sta97A] is the fundamental protocol on the **Internet.** It provides connectionless, packet-based communication. Industrial automation has long been a good application area for Internet-based embedded systems. Information appliances that use the Internet are rapidly becoming another use of IP in embedded computing. The term *Internet* generally refers to the global network of

computers connected by the IP. However, it is possible to build an isolated network not connected to the global Internet that uses IP.

Internetworking

IP is not defined over a particular physical implementation; it is an **internetworking** standard. Internet packets are assumed to be carried by some other network, such as Ethernet. Generally, an Internet packet travels over several different networks from source to destination. The IP allows data to flow seamlessly through these networks from one end user to another. The relationship between IP and individual networks is illustrated in Fig. 8.5. IP works at the NWK layer. When node *A* wants to send data to node *B*, the application's data pass through several layers of the protocol stack to get to the Internet Protocol, which then creates packets for routing to the destination. These are then sent to the *data link* and *PHY* layers. A node that transmits data among different types of networks is known as a **router**. The router's functionality must go up to the IP layer, but because it does not run applications, it does not need to go to higher levels of the OSI model. In general, a packet may go through several routers to reach its destination. At the destination, the IP layer provides data to the transport layer and ultimately to the receiving application. As the data pass through several layers of the protocol stack, the IP packet data are encapsulated in packet formats appropriate to each layer.

IP packets

The basic format of an IP packet is shown in Fig. 8.6. The header and data payload are both variable in length. The maximum total length of the header and data payload is 65,535 bytes.

An Internet address is a number (32 bits in early versions of IP, 128 bits in IPv6). The IP address is typically written in the form xxx.xxx.xxx.xxx. The names by which users and applications typically refer to Internet nodes, such as foo.baz.com, are translated into IP addresses via calls to a **Domain Name Server** (**DNS**), one of the higher-level services built on top of IP.

The fact that IP works at the network layer tells us that it does not guarantee that a packet is delivered to its destination. Furthermore, packets that do arrive may come out of order. This is referred to as **best-effort routing**. Because routes for data may change quickly with subsequent packets being routed along different paths with

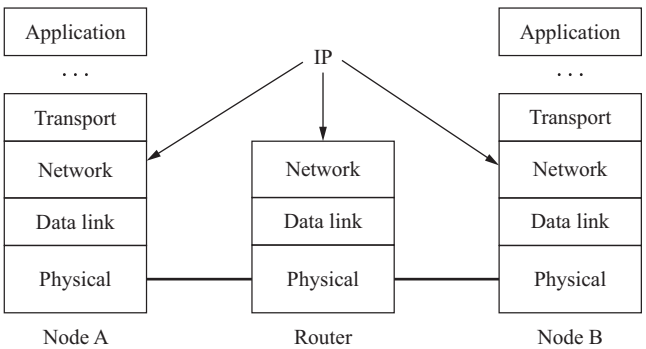


FIGURE 8.5

Protocol utilization in Internet communication.

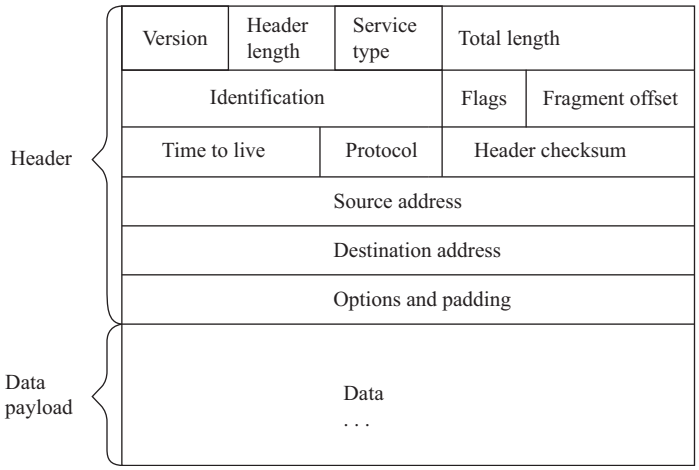


FIGURE 8.6

Internet protocol packet structure.

different delays, the real-time performance of IP can be hard to predict. When a small network is contained totally within the embedded system, performance can be evaluated through simulation or other methods because the possible inputs are limited. Because the performance of the Internet may depend on worldwide usage patterns, its real-time performance is inherently harder to predict.

IP services

The Internet also provides higher-level services built on top of IP. The **Transmission Control Protocol (TCP)** is one such example. It provides a connection-oriented service that ensures that data arrive in the appropriate order, and it uses an acknowledgment protocol to ensure that packets arrive. Because many higher-level services are built on top of TCP, the basic protocol is often referred to as TCP/IP.

Fig. 8.7 shows the relationships between IP and higher-level Internet services. Using IP as the foundation, TCP is used to provide **File Transport Protocol** (**FTP**) for batch file transfers, Hypertext Transport Protocol (**HTTP**) for World Wide Web service, **Simple Mail Transfer Protocol** (**SMTP**) for email, and Telnet for virtual

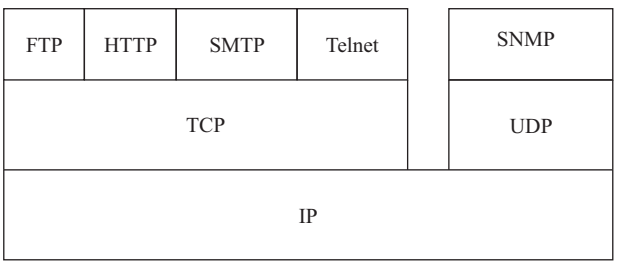


FIGURE 8.7

Internet service stack.

terminals. A separate transport protocol, the **User Datagram Protocol** (**UDP**), is used as the basis for the network management services provided by the **Simple Network Management Protocol** (**SNMP**).

8.4.3 IoT networking concepts

Not everything is connected to the Internet. Although the Internet is a good match for computer systems, it is not always well suited to other types of devices. Many IoT devices communicate over non-IP networks, which are sometimes called **edge networks**. As shown in Fig. 8.8, devices can link to the Internet using a **gateway** that translates between the IoT network and the Internet.

Ad hoc networks

IoT networks differ from traditional Internet in that they do not need to be explicitly set up and managed. An **ad hoc network** is created by the self-organization of a set of nodes. The nodes route messages to each other; they do not rely on separate routers or other networking equipment.

We can evaluate an IoT network based on both its functional and nonfunctional characteristics:

- Does it provide adequate security and privacy?
- How much energy is required for communication? Many IoT network devices are designed to operate from a button battery for an extended period. We refer to these networks as **ultra-low energy** (**ULE**).
- How much does the network cost to add to a device?

Services

An ad hoc network should supply several services:

- **Authentication** determines whether a node is eligible to be connected to the network.

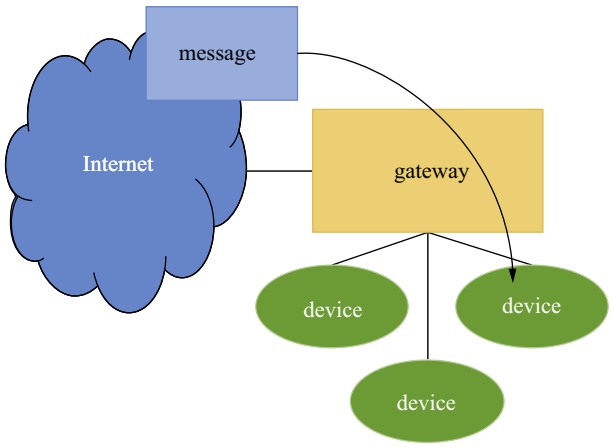


FIGURE 8.8

Gateway between the Internet and a personal area network.

- **Authorization** checks whether a given node should be able to access a piece of information on the network.
- **Encryption and decryption** help provide security.

The **topology** of a network describes the structure of communication within the network. In the OSI model, a link is a direct connection between two nodes. Communication between two nodes that are not directly connected requires several hops. The topology of a wireless IoT network is influenced by several factors, including the range of radios and the complexity of the management of the topology. Fig. 8.9 shows several examples of IoT network topologies. The **star network** uses a central hub through which all other nodes communicate. A **tree network** provides a more complex structure, but still only provides one path between a pair of nodes. A **mesh network** is a general structure.

Once the network authenticates a node, it must perform housekeeping functions to incorporate a new node into the network. **Routing discovery** determines the routes that will be used by packets that travel to and from other nodes to the new node. Routing discovery starts by searching the network for paths to the destination node. A node will broadcast a message requesting routing discovery services and record the response it receives. The recipient nodes will then broadcast their own routing discovery request, with the process continuing until the destination node is reached. Once a set of routes has been identified, the network evaluates the cost of choosing one (or perhaps more than one) path. The cost computation for a path can include the number of hops, the transmission energy required, and the signal quality on each link.

Routing discovery produces a routing table for each node, as illustrated in Fig. 8.10. When a node wants to send a message to another node, it consults its routing table to determine the first node on the path. That node consults its own routing table to determine the next hop; the process continues until the packet reaches its destination.

QoS

Many IoT networks support **synchronous** and **asynchronous** communication. Synchronous communication is periodic, for example, voice or sampled data. We often use the term **quality-of-service** (**QoS**) to describe the bandwidth and periodicity characteristics of synchronous data. To provide synchronous data with QoS

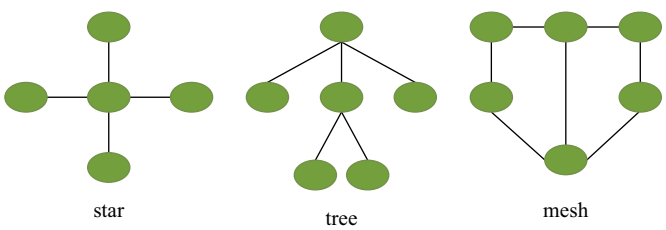


FIGURE 8.9

Example of IoT network topologies.

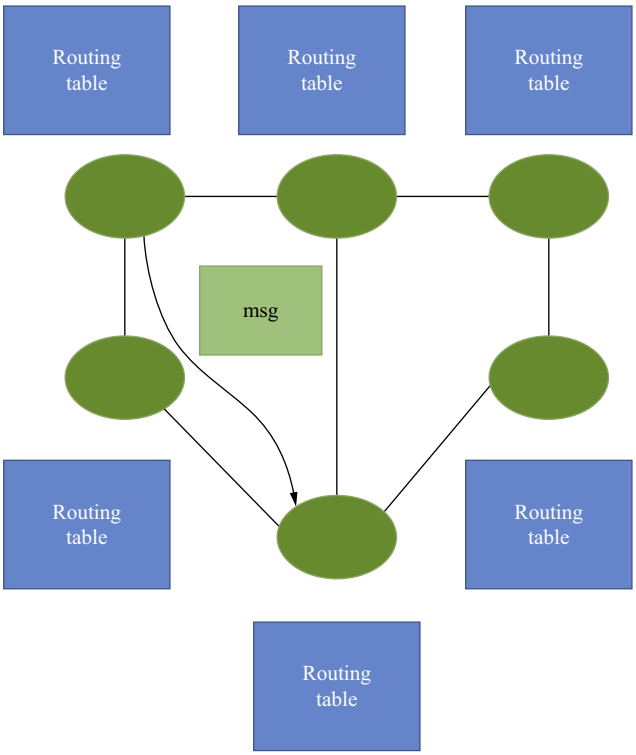


FIGURE 8.10

Routing packets through a network.

characteristics, the network needs to reserve bandwidth for that communication. Many networks perform **admission control** to process a request for synchronous transmission and to determine whether the network has the bandwidth available to support the request. A request may be rejected if, for example, too many existing synchronous communication streams do not leave sufficient bandwidth to support the requested connection.

Synchronization

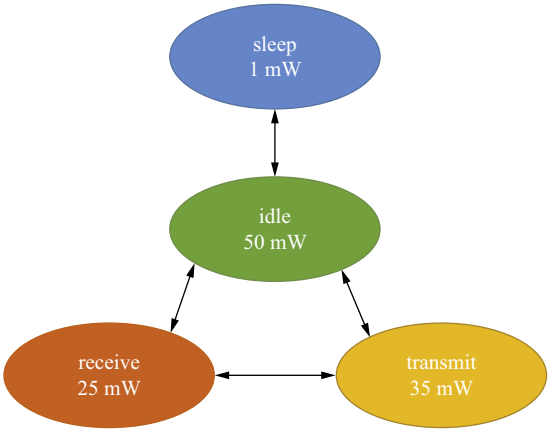
One challenge in synchronous communication over wireless networks is synchronizing the nodes. Many wireless networks provide synchronous communication using **beacons**. As illustrated in Fig. 8.11, the beacon is a transmission from a node that marks the beginning of a communication interval. The time between beacons is usually divided into two segments, one for synchronous packets and the other for asynchronous packets. A synchronous communication can be assigned its own time slot in the synchronous segment.

The energy required for communication is a key concern for wireless battery-operated networks. Energy requirements may be stated in two styles: either Joules or Watts. Charge is expressed in amp-hours. The amp-hour metric can be converted to energy using the power supply voltage.

FIGURE 8.11

Beacon transmissions.

Energy consumption is generally evaluated for particular use cases that consider idle time, transmission length, or other factors. We can use the power state machine in Section 3.7.2 to help us compute wireless energy consumption. A use case describes a path through a power state machine. As shown in Fig. 8.12, given the time spent in each state and the power consumption in those states, we can compute the energy consumption for a use case.



A

step	state	time	energy
1	sleep	1 ms	1 nJ
2	idle	10 ms	0.5 nJ
3	receive	50 ms	1.25 nJ
4	transmit	50 ms	1.75 nJ
5	receive	50 ms	1.25 nJ
6	transmit	50 ms	1.75 nJ
			total = 6 nJ

B

FIGURE 8.12

Example of radio energy consumption analysis. (a) Radio power state machine. (b) Use-case-based energy analysis.

8.4.4 Bluetooth and Bluetooth Low Energy

Bluetooth was introduced in 1999, originally for telephony applications, such as wireless headsets for cell phones. It is now used to connect a wide range of devices to host systems. **Bluetooth Low Energy** (**BLE**) shares a name but has a quite different design.

Classic Bluetooth

Classic Bluetooth [Mil01], as the original standard has come to be known, is designed to operate in a radio band known as the **instrumentation, scientific, and medical** (**ISM**) **band**. The ISM band is in the 2.4-GHz frequency range and no license is required to operate in the ISM band throughout the world. There are, however, some restrictions on how it can be used, such as the 1 MHz bandwidth channels and frequency-hopping spread spectrum.

Bluetooth networks are often called **piconets** thanks to their small physical size. A piconet consists of a master and several slaves. A slave can be active or parked. A device can be a slave on more than one piconet.

The Bluetooth stack is divided into three groups: **transport protocol**, **middleware protocol**, and **application**.

The transport protocol group has several constituents:

- The radio provides physical data transport.
- The baseband layer defines the Bluetooth air interface.
- The link manager performs device pairing, encryption, and negotiation of link properties.
- The **LLC and adaptation protocol** (**L2CAP**) layer provides a simplified abstraction of transport for higher levels. It breaks large packets into Bluetooth packets. It negotiates the QoS required and performs admission control.

The middleware group has several members:

- The RFCOMM layer provides a serial port-style interface.
- The service discovery protocol (SDP) provides a directory for network services.
- IP and IP-oriented services, such as TCP and UDP.
- A variety of other protocols, such as IrDA for infrared and telephony controls.

The application group includes the various applications that use Bluetooth.

Every Bluetooth device is assigned a 48-bit Bluetooth device address. Every Bluetooth device also has its own Bluetooth clock that is used to synchronize the radios on a piconet, as required for frequency-hopping spread spectrum communication. When a Bluetooth device becomes a part of a piconet, it adjusts its operation to the clock of the master.

Transmissions on the network alternate between master and slave directions. The baseband supports two types of packets:

- **Synchronous connection-oriented** (**SCO**) packets are used for QoS-oriented traffic, such as voice and audio.

- **Asynchronous connectionless** (**ACL**) packets are used for non-QoS traffic.

SCO traffic has a higher priority than ACL traffic.

Bluetooth Low Energy

BLE [Hey13] is, as the name implies, designed to support very low energy radio operation. A radio operated by a button-sized battery for an extended period is an example scenario of BLE usage. BLE is part of the Bluetooth standard, but it differs in some fundamental ways from Classic Bluetooth. For example, BLE uses a different modulation scheme in the PHY layer than does Classic Bluetooth. BLE does, however, share some features and components of Classic Bluetooth, such as the L2CAP layer.

Minimizing the amount of time the radio is on is critical to low-energy operation. BLE is designed to minimize radio on-time in several ways. At the link level, packets are designed to be relatively small. BLE is also designed to support communications that do not require long-lived connections.

Advertising is one form of communication that is designed to support low-energy operation. A device can transmit advertising packets; devices can also listen to advertising packets. Advertising can be used to discover devices or broadcast information. Some short communications may be possible entirely through advertising. If longer communications are required between devices, BLE also supports the establishment of connections.

Fig. 8.13 shows the BLE link-level state machine. The scanning state allows the device to listen to advertising packets from other devices: passive scanning only listens, while active scanning may also send requests for additional information. The advertising state corresponds to a device transmitting advertising packets. A connection may be entered through either the initiating or advertising states. As with Classic Bluetooth, a device in a connection is either a master or a slave. The standby state is reachable from all other states.

The BLE Host Controller Interface (HCI) provides several interfaces to the host: UART provides simple communication facilities; three-wire UART adds capabilities to UART for link establishment and acknowledgment; USB provides high-speed communication; and Secure Digital Input Output (SDIO) is designed for medium-speed communication.

The **Attribute Protocol Layer** provides a mechanism to allow devices to create application-specific protocols for their particular data communication and management needs. An **attribute** has three components: its **handle**, which functions as the name of the attribute; its type, which is chosen from the set of **Universally Unique Identifiers** (**UUIDs**); and its **value**. Most UUIDs used in BLE devices are part of a restricted set built around the **Bluetooth-based UUID** and come in several types: service UUIDs, units, attribute types, characteristic descriptors, and characteristic types.

Attributes are collected in an **attribute database** maintained in an attribute server. An attribute client can use the **Attribute Protocol** to query a database. Each device has only one attribute database. An attribute has permissions: readable, writable, or both. Attributes can also be protected with authentication and authorization.

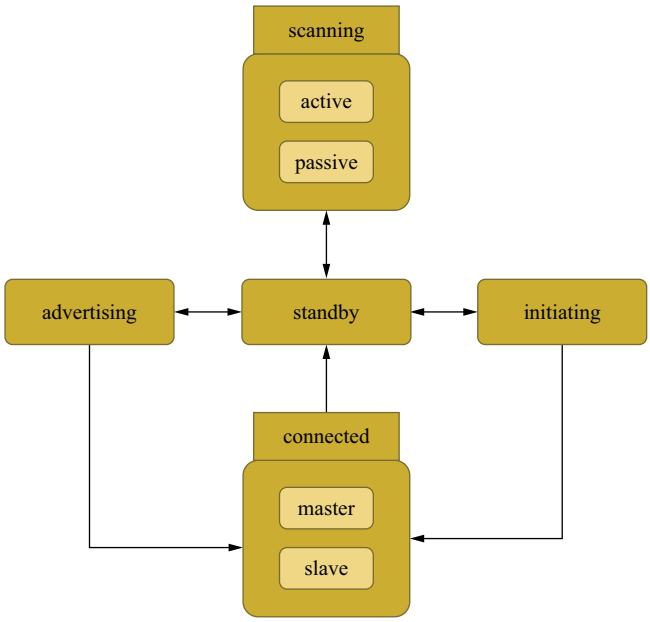


FIGURE 8.13

Bluetooth Low Energy link-level state machine.

A set of attributes can be used to define a state machine for a protocol. The states and transitions in the state machine can be stored as attributes; the current state, inputs, and outputs can also be represented by attributes.

The **Generic Attribute Profile Layer** (GATT) defines a basic set of attributes for all BLE devices. The main purpose of the Generic Attribute Profile is to define the procedures for discovery and for the interactions of clients and servers.

BLE provides mechanisms for security. Two devices communicating for the first time is known as **pairing**. This process uses a **short term key** to send a **long term key** to the device. The long-term key is stored in the database, a process known as **bonding**. Data sent as part of a connection may be encrypted using the advanced encryption standard.

8.4.5 802.15.4 and ZigBee

ZigBee is a widely used personal area network (PAN) based on the IEEE 802.15.4 standard (sorry, this standard has no catchy name). 802.15.4 defines the MAC and PHY layers; ZigBee builds upon those definitions to provide several application-oriented standards.

802.15.4 [IEE06] can operate in several different radio bands, including ISM, but also others. The standard is designed for systems with either no battery or those that allow only a small current draw from the battery.

802.15.4 supports two types of devices: **full-function devices** (**FFDs**) and **reduced-function devices** (**RFDs**). An FFD can serve as either a device, a coordinator, or a personal area network coordinator. An RFD can only be a device. Devices can form networks using either a star topology or a peer-to-peer topology. In the case of a star topology network, a PAN coordinator serves as a hub; peer-to-peer networks also have a PAN coordinator, but communications do not have to go through the PAN coordinator.

The basic unit of communication in an 802.15.4 network is the frame, which includes addressing information, error correction, and other information, as well as a data payload. A network can also make use of optional superframes. A superframe, which is divided into 16 slots, has an active portion, followed by an inactive portion. The first slot of a superframe is known as the **beacon**. It synchronizes the nodes in the network and carries network identification information. To support QoS guarantees and low-latency operations, the PAN coordinator may dedicate parts of a superframe to those high-QoS or low-latency operations. Those slots do not have contention.

The PHY layer activates the radio, manages the radio link, sends and receives packets, and other functions. It has two major components: the PHY data service and the PHY management service. The interface to the PHY layer is known as the *physical layer management entity service access point* (*PLME-SAP*). The standard uses *carrier sense multiple access with collision avoidance* (*CSMA-CA*).

The MAC layer processes frames and other functions. It also provides encryption and other mechanisms that can be used by applications to provide security functions. The MAC layer consists of the MAC data service and MAC management service; its interface is known as the *MLME-SAP*.

ZigBee [Far08] defines two layers above the 802.15.4 PHY and MAC layers: the **NWK** layer provides network services, and the **APL** layer provides application-level services.

The ZigBee NWK layer forms networks, manages the entry and exit of devices to and from the network, and manages routing. The NWK layer has two major components. The **NWK Layer Data Entity** (**NLDE**) provides data transfer services, and the **NWK Layer Management Entity** (NLME) provides management services. A **Network Information Base** (**NIB**) holds a set of constants and attributes. The NWK layer also defines the network address for the device.

The NWK layer provides three types of communication: broadcast, multicast, and unicast. A broadcast message is received by every device on the broadcast channel. Multicast messages are sent to a set of devices. A unicast message, the default type of communication, is sent to a single device.

The devices in a network may be organized in many different topologies. A network topology may be determined, in part, by which nodes can physically communicate with each other, but the topology may be dictated by other factors. A message may, in general, travel through multiple hops in the network to its destination. A ZigBee coordinator or router performs a routing process to determine the route through a network used to communicate with a device. The choice of a route can be guided by

several factors: the number of hops or link quality. The NWK layer limits the number of hops that a given frame is allowed to travel.

The ZigBee APL layer includes an **application framework**, an **application support sublayer** (**APS**), and a **ZigBee Device Object** (**ZDO**). Several **application objects** may be managed by the application framework, each for a different application. The APS provides a services interface from the NWK layer to the application objects. The ZDO provides additional interfaces between the APS and the application framework.

ZigBee defines several **application profiles** that define a particular application. The ZigBee Alliance issues the **application identifier**. The application profile includes a set of **device descriptions** that provide the characteristics and state of the device. One element of the device description also points to a **cluster** that consists of a set of attributes and commands.

8.4.6 Wi-Fi

The 802.11 standard, known as Wi-Fi [IEE97], was originally designed for portable and mobile applications such as laptops. The original standard has been extended several times to include higher-performance links in several different bands. It was designed before ultra-low-energy networking became an important goal. However, a new generation of Wi-Fi designs is designed for efficient power management and operates at significantly lower power levels.

Wi-Fi supports ad hoc networking. A **basic service set** (**BSS**) consists of two or more 802.11 nodes that communicate with each other. A **distribution system** (DS) interconnects BSSs. More expansive links are provided by an **extended service set** (**ESS**) network. BSS related by an ESS can overlap or be physically separate. A **portal** connects the wireless network to other networks.

A network provides a set of services. The most basic service is the **distribution** of messages from source to destination. **Integration** delivers a message to a portal for distribution by another network. **Association** refers to the relationship of a station to an access point; **reassociation** allows an association to be moved to a different access point; and **disassociation** allows an association to be terminated. Every station must provide authentication, deauthentication, privacy, and MAC service data unit (MSDU) delivery. A DSS must provide association, disassociation, distribution, integration, and reassociation.

The reference model for 802.11 breaks the PHY layer into two sublayers: **PHY Layer Convergence Protocol** (**PLCP**) and **Physical Medium Dependent** (**PMD**). They communicate with a PHY sublayer management entity. The MAC sublayer communicates with a MAC sublayer management entity. Both management entities communicate with the station management entity.

MAC provides several services:

- Asynchronous data service. This service is connectionless and best-effort.

- Security. Security services include confidentiality, authentication, and access control.
- StrictlyOrdered service. A variety of effects can cause frames to arrive out of order. This service ensures that higher levels see the frames in the strict order in which they are transmitted.

The next example describes a low-power Wi-Fi device.

Example 8.2: Qualcomm QCA4004 Low-Power Wi-Fi

The QCA4004 [Qua15] is a Wi-Fi device designed for low-energy operation. It operates in both 2.4- and 5-GHz bands. Low-energy features include power-saving modes with fast wakeup times. The chip can be interfaced with devices using GPIO or I^2C.

8.4.7 LoRa

LoRa stands for *long range*. This network is designed for both low-power operations suitable for IoT systems and coverage over wide areas. The term *LoRa* primarily refers to the PHY layer, which is based on a form of spread spectrum. LoRaWAN is a protocol developed to take advantage of the LoRa PHY layer. An asynchronous protocol is used to allow devices to send data when required, while reducing the power consumption for idle intervals.

8.5 Databases and timewheels

In this section, we consider mechanisms that can be used to organize information in IoT systems. **Databases** are used in many applications to store collections of information. Since IoT systems often operate devices in real time, a **timewheel** can be used to manage the temporal behavior of the system.

8.5.1 Databases

IoT networks use databases to manage and analyze data from IoT devices. IoT databases are often kept in the cloud; this is particularly true when we want to not just store data, but also compute on it. To understand how to use databases, we first need to consider how to put data into the database and then how to extract data from the database.

Relational database

The traditional database model is the **relational database management system** (**RDBMS**) [Cod70]. The term relational comes from mathematics: a relation is a Cartesian product of a set of domain values with a set of range values.

Data representation

As shown in Fig. 8.14, data in a relational database are organized into tables. The rows of the table represent **records** (sometimes called **tuples**). The columns of the table are known as **fields** or **attributes**. One column of the table (or sometimes a set of

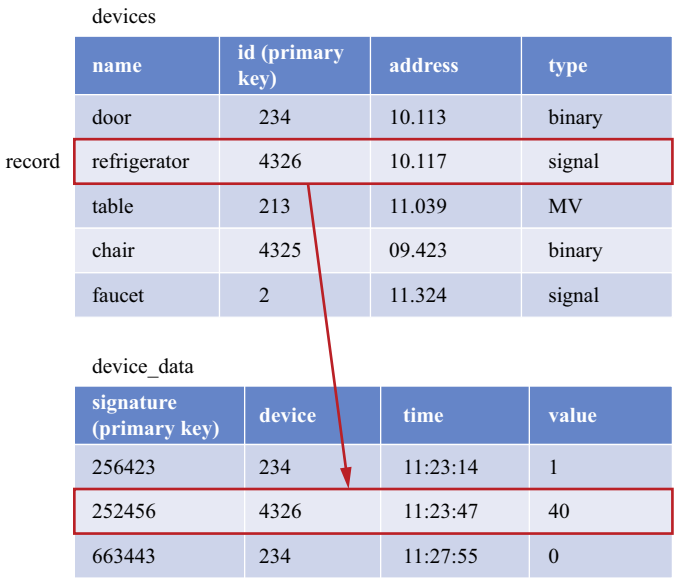


FIGURE 8.14

Tables in a database.

columns) is used as a **primary key**; each record has a unique value for its primary key field, and each key value uniquely identifies the values of the other columns. The *devices* table in the example defines a set of devices. The *id* field serves as the primary key for the devices. The *device_data* table records timestamped data from the devices. Each of these records has its own primary key, given the name *signature*. Because those records also contain the *id* for the device that recorded the data, we can use a device's *id* field to look up records in the *device_data* table.

A database contains, in general, more than one table. The set of table definitions in a database is known as its **schema**. *Requirements analysis* is the process of determining what data are required for a given application, as in object-oriented program design.

From a logical perspective, eliminating redundancy is key to maintaining the data in the database. If a piece of data is stored in two different tables (or two different columns in one table), then any change to the data must be recorded in all of its copies. If some copies of the data are changed and some are not, then the value returned will depend on which copy was accessed. In practice, redundant values may improve access times; database management systems can perform this type of optimization without requiring database designers to use redundant schemas.

The database designer's description of the tables does not necessarily reflect how the data are organized in memory or on a disk. The database management system may perform some optimizations to reduce storage requirements or improve access speed. The relational model does not order the records in the table, which helps to

give the database management system more freedom to store the data in the most efficient format.

Normal forms

Database **normal forms** are rules that help us create databases without redundancies and other types of problems that may cause problems in database management. Many different normalization rules have been created, some of which we consider here. The **first normal form** is a schema in which every cell contains only a single value; every record has the same number of fields.

The **second normal form** obeys the first normal form, and the values of all the other cells in a record are unique to the key. If a database does not obey the second normal form, then we have duplicated the information in the database. For example, an example database in the second normal form database has two tables: the records in one table contain sensor name, network address, and physical location; records in another table contain sensor name, read time, and value. The name/time pair forms a key for the second table. If the second table's records also included the sensor's physical location, it would not be in second normal form because the sensor location does not depend on the name/time of a reading. Updating the physical location of a sensor would require updating multiple records.

The **third normal form** satisfies the second normal form (and therefore also the first); it also requires that the non-key columns be independent. A database in third normal form has two tables: one with sensor name and sensor model number and another with sensor model number and sensor type (motion, video, and so forth). If the database were modified to have a single record with sensor name, model number, and type, it would not be in the third normal form because the type can be inferred from the model number.

Queries

A request for information is known as a **query**. Users do not deal directly with tables. Instead, they formulate a request in a **query** language known as **structured query language** (**SQL**). The result is the set of records that satisfy the query. A query may result in more than one record. In the example of Fig. 8.14, we can ask for all the *device_data* records for the door using the query:

```
select from device_data where device = 234
```

The result would be two records.

Joins

A common type of query combines information from more than one table. This operation is known as a **join**. A join can be described mathematically as a Cartesian product of rows. Tables can be related to each other in several different ways:

- one-to-one, if a record in one table corresponds to exactly one record in another table, for example, a sensor and its network address;
- one-to-many, in which a record in one table is related to many records in another table, such as a sensor and a set of readings; and
- many-to-many, in which a set of records can be related to another set of records, such as a group of sensors and readings from that sensor group.

The database management system's job is to efficiently perform the logical operations described by the join. If one type of query is known to occur frequently, the

database management system may optimize the internal representation for that type of query; these sorts of optimizations do not change the schema but merely represent the data in a particular format that is hidden from the user.

Other types of relationships beyond join are possible. A **projection** eliminates some columns in a relation, for example, by paring down a lengthy record of a person to the fields requested by a query. A **restriction** eliminates some rows from a table, for example, by returning only fields with a last name, starting with "A."

Schemaless databases

An alternative to the relational model is the **schemaless** or **noSQL** database. The term *schemaless* is something of a misnomer; the data are stored as records, but there is no central knowledge of the format of those records. Data are stored in collections of arrays or tables. The database designer writes software methods to access and modify database records. **JavaScript Object Notation** (**JSON**) [ECM13] is often used to describe records in a schemaless database. JSON syntax builds objects out of two basic component data structures: name/value pairs and ordered lists of values.

8.5.2 Timewheels

The timewheel allows the IoT system to process events in the order in which they occur. This is particularly important when controlling devices—turning lights on and off at specified times, for example. Timewheels are used in event-driven simulators to control the order in which simulated events are processed. We can also use timewheels to manage the temporal behavior of devices in an IoT system [Coe14].

As shown in Fig. 8.15, the timewheel is a sorted list of input and output events. As input events arrive, they are put in sorted order into the queue. Similarly, when output events are scheduled, they are placed in the queue at the proper time order.

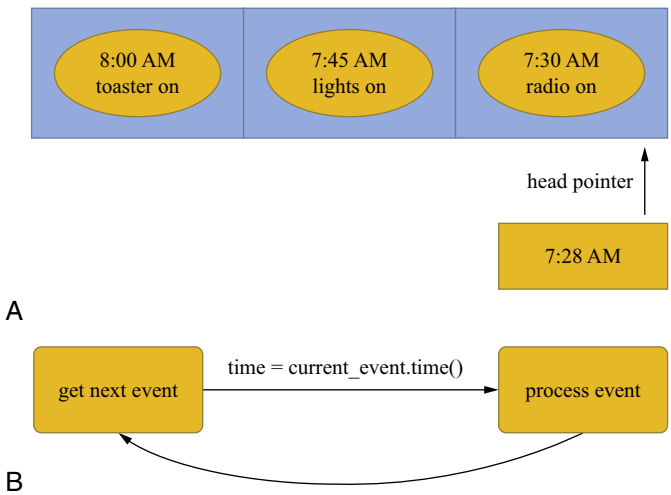


FIGURE 8.15

Timewheel organization. (a) Timewheel queue. (b) Unified modeling language state diagram.

Fig. 8.15 also provides a UML state diagram for the operation of the timewheel. It pulls the head event from its queue. When the time specified in the event is reached, that event is processed.

A system may have one or more timewheels. A central timewheel can be used to manage activity throughout the entire IoT system. In larger networks, several timewheels can be distributed around the network, each of which keeps track of local activity.

8.6 Example: smart home

A **smart home** is a house equipped with sensors that monitor activity and help run the house. A smart home may provide several types of services:

- remote or automatic operation of lights and appliances;
- energy and water management for efficient use of natural resources; and
- monitoring the activities of residents.

Smart homes can be particularly helpful to residents, such as senior citizens or people with special needs [Wol15]. The smart home can analyze activities to, for example, be sure that the resident is taking care of daily tasks. It can also provide summaries of the resident's activity to loved ones, caregivers, and health care professionals. Performing these tasks requires not only operating sensors, but also analyzing the sensor data to extract events and patterns. A smart home system can provide three types of outputs:

- reports on the activities of residents;
- alerts for out-of-the-ordinary activity; and
- recommendations to the residents and caregivers as to what actions may be taken.

Fig. 8.16 shows the layout of a typical smart home. The home includes several cameras to monitor common areas. Other types of sensors can also help to keep track of activity: door sensors tell when someone has passed through a door, but does not give the person's identity or even whether they are entering or leaving; sensors on water faucets can tell when someone is using a bathroom or kitchen sink; and electrical outlet sensors can tell when someone is using an electrical appliance. Appliances and devices can also be controlled: lights can be turned on and off; heaters and air conditioners can be managed; sprinklers can be turned on and off; and so on.

Fig. 8.17 shows how sensors can be used to monitor a resident's activity. The resident leaves her bedroom, walks through the hallway to the bathroom, uses the sink, then moves to the living room, and turns on the TV. Sensors can be used to monitor these actions. In the case of the hall camera, computer vision algorithms can identify the person in the hallway and track their movement from one room to another. However, the other sensors provide only indirect information. Analysis algorithms can use

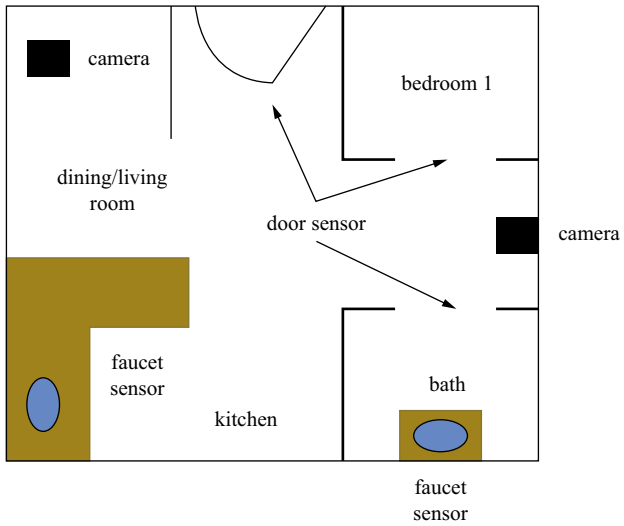


FIGURE 8.16

IoT network in a smart home.

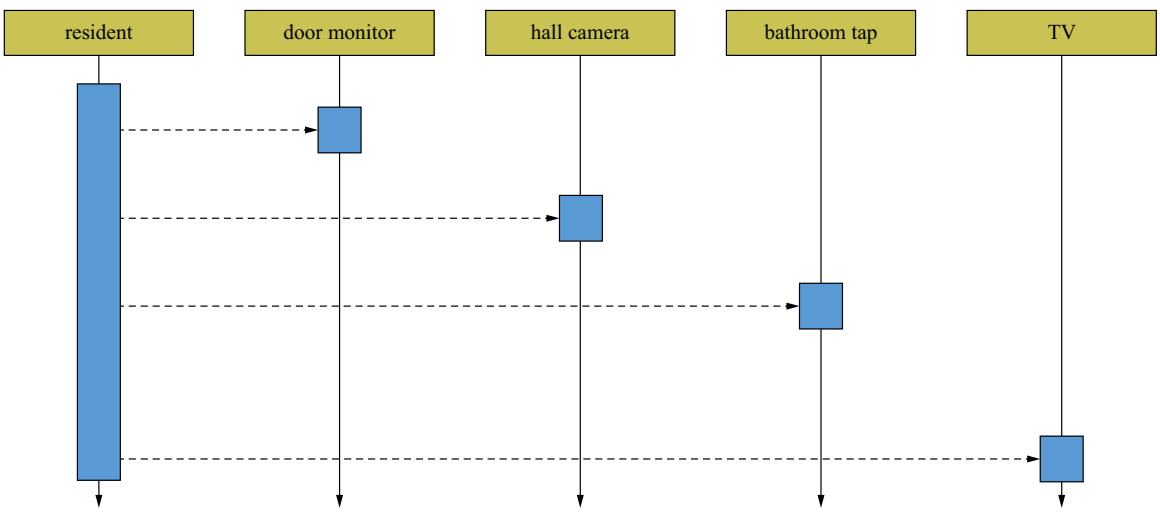


FIGURE 8.17

Analyzing a resident's activity in a smart home.

statistical methods to infer that the same person was likely to have caused all these events. For example, a person in another part of a house may not be able to change locations quickly enough to cause some of the events.

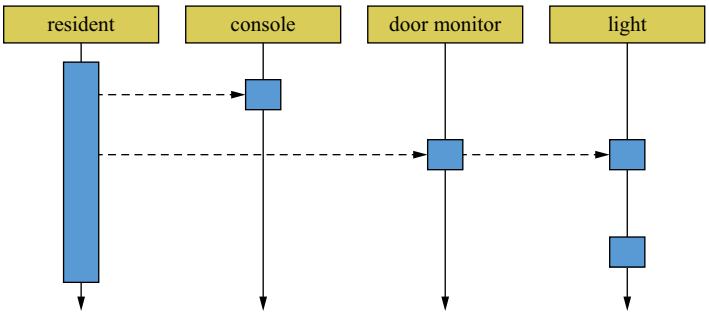


FIGURE 8.18

Light activation in a smart home.

Fig. 8.18 shows how the smart home can control and monitor activities. The resident uses the console to set up two conditions in which the light should be turned on: whenever anyone goes through the resident's doorway, or at a specified time. In this use case, the resident later goes through the doorway, causing the light to turn on for a specified interval. The light also goes on automatically at the appointed time.

Fig. 8.19 shows an object diagram of a smart home. Sensors and hubs form the network. The console is the main UI. Data are processed in two stages. Sensor readings and events are initially processed by a timewheel that manages the timely operation of devices in the house. Not all sensor readings are of long-term interest. Sensor events for long-term analysis are passed to a database. A database can reside in the cloud and allow access to a variety of analysis algorithms.

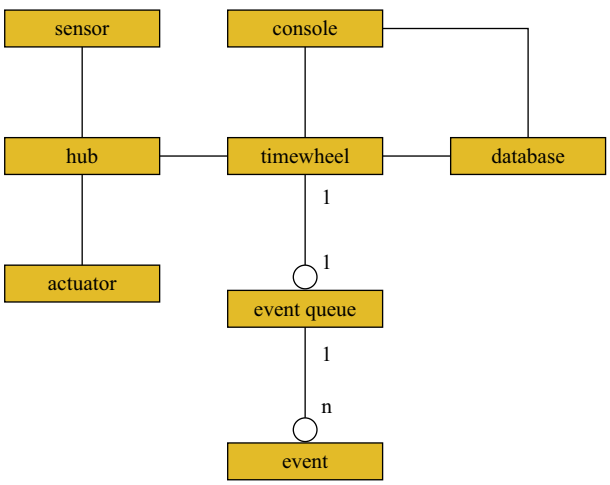


FIGURE 8.19

UML object diagram for a smart home.

8.7 Summary

IoT systems leverage low-cost, network-enabled devices to build sophisticated networks. Some different networks can be used to build IoT systems; many practical systems combine several networks. Databases are often used to manage information about IoT devices. Timewheels can be used to manage events in the system.

What we learned

- An IoT system connects edge devices into a network and manages information about those devices in soft real time.
- Bluetooth, ZigBee, and Wi-Fi are all used to connect IoT devices to the rest of the network.
- Databases can be used to store and manage information about IoT devices. Timewheels can be used to manage time-oriented activities in the network.

Further reading

Karl and Willig discuss wireless sensor networks [Kar06]. Serpanos and Wolf discuss IoT systems [Ser18]. Farahani [Far08] describes ZigBee networks. Heydon [Hey13] describes BLE networks.

Questions

Q8-1 Classify these Bluetooth layers using the OSI model:

a) baseband
b) L2CAP
c) RFCOMM

Q8-2 Use the power state machine in Fig. 8.12 to determine the energy used in these use cases:

a) Idle 1 s; receive 10 ms; idle 0.1 s; transmit 5 µs.
b) Sleep 1 min; receive 50 ms; idle 0.1 s; receive 100 ms.
c) Sleep 5 min; transmit 5 µs; receive 10 ms; idle 0.1 s; transmit 10 µs.

Q8-3 Design the schema for a database table that records the activation times of a motion sensor.

Q8-4 Design the schema for a single database table that records the activation times for several motion sensors.

Q8-5 You are given a timewheel that is initially empty. The timewheel processes events, each of which has a generation time and a release time. Show the state of the timewheel (events and their order) after each of these events is received at its generation time. Times are given as mm:ss (minutes, seconds).

a) e1: generation 00:05, release 00:06.
b) e2: generation 00:10, release 20:00.
c) e3: generation 01:15, release 10:00.
d) e4: generation 12:15, release 12:20.
e) e5: generation 12:16, release 12:18.

Lab exercises

L8-1 Use Bluetooth to connect a simple sensor, such as an electric eye, to a database.

L8-2 Use a temperature sensor and a motion sensor to determine the average temperature in a room when a person is present.

L8-3 Design a database schema for a smart classroom. Identify the features of the smart classroom and design the schema to support those use cases.