# Chapter 5
# IoT Protocol Stack: A Layered View


The IoT protocol stack can be visualized as an extension of the TCP/IP layered protocol model and is comprised of the following layers (refer to Fig. 5.1):

- Physical layer
- Link layer
- Network layer
- Transport layer
- Application Protocols layer
- Application Services layer

Note that the Application layer of the TCP/IP protocol stack is expanded into two layers in the IoT protocol stack: Application Protocols and Application Services. It is as if the proverbial "narrow waist" of the hourglass is being extended further up the stack to provide interoperability between heterogeneous "things."

## 5.1 Link Layer

In this section we will examine the impact of the IoT requirements on the Link layer through a combined view of the challenges that those requirements impose on networking technologies, industry efforts to address those challenges, and remaining gaps.

### *5.1.1 Challenges*

The challenges that the IoT presents to the Link layer of the protocol stack can be broadly categorized into the following four areas: device characteristics, traffic characteristics, access characteristics, and scalability (Fig. 5.2).



A. Rayes, S. Salam, *Internet of Things From Hype to Reality*,
https://doi.org/10.1007/978-3-319-99516-8_5

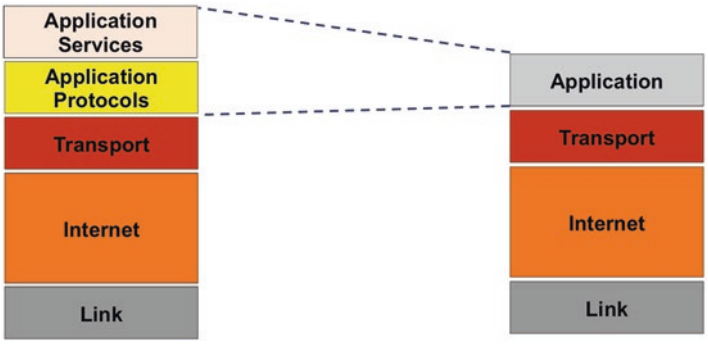



**Fig. 5.1** IoT protocol stack

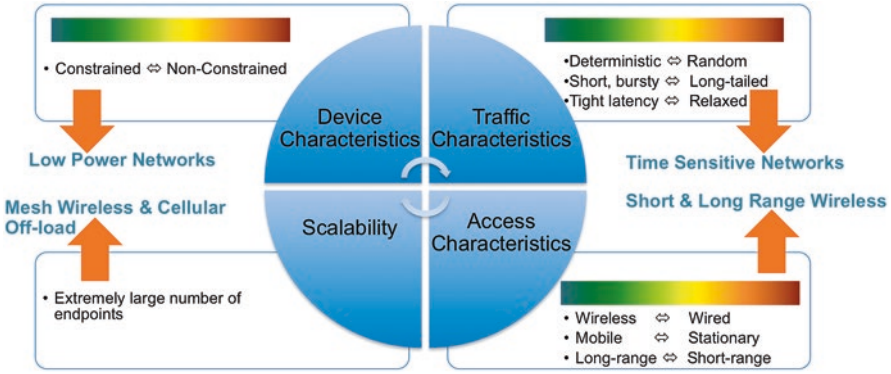



**Fig. 5.2** Link layer challenges. (Source Cisco BRKIOT-2020, 2015)

On the device characteristics front, the IoT will encompass a wide spectrum of "things" that span from fully capable (non-constrained) compute nodes to highly constrained devices. The latter typically have limited energy resources to spend on processing and communication. As discussed earlier, network communication is typically more power consuming when compared to local processing. Hence, communication technologies need to be optimized to accommodate low-power devices. Implementation of protocols at all layers of the protocol stack can affect energy consumption. However, the Link layer, in particular, has a significant impact due to the fact that this layer is responsible for the nuances of the physical transmission technology, framing, media access control, and retransmissions. For instance, it is reported that, depending on the link load, between 50% and 80% of the communication energy is used for repairing lost transmissions at the MAC layer.

The traffic characteristics of IoT endpoints vary widely depending on the application's demands and nature of devices. Some applications have relaxed require-

ments on packet loss, latency, and jitter (e.g., a meteorological monitoring application), whereas others have very tight availability, latency, and jitter tolerance (e.g., a jet engine control application). It is worth noting here the contrast between the meteorological monitoring and jet engine control applications: both applications may be using the same types of devices (temperature sensors, pressure sensors) and observing the same physical entities (temperature, pressure). However, it is the applications' requirements that dictate the traffic characteristics that the network must deliver. By the same token, some IoT devices generate short bursty traffic (e.g., point of sale terminal), whereas other devices generate long-tailed traffic (e.g., video camera). The dichotomy in traffic characteristics, between solutions that expect determinism and those that can withstand best-effort (random) communications, creates drivers for Link layer technologies that support deterministic and Time-Sensitive Networking.

The access characteristics of IoT endpoints become increasingly diverse as the footprint of the network grows beyond traditional IT environments, dominated by familiar local area network (LAN) and wide area network (WAN) technologies, and into new deployment environments such as industrial plant floor, oil fields, marine platforms, mines, wells, power grids, vehicles, locomotives, and even the human body. IoT devices in these environments may connect to the network using a mix of wireless and wired technologies. The devices when connected wirelessly may be either mobile or stationary and depending on the logistics of the deployment may require either long-range or short-range connectivity solutions. To accommodate this diversity, new Link layer protocols that form the foundation of field area network (FAN), neighborhood area network (NAN), and personal area network (PAN) technologies are required.

IoT scalability demands present interesting challenges for the Link layer of the protocol stack, especially for wireless technologies. On the one hand, these technologies offer a number of appealing characteristics that make them a good fit for the IoT, low upfront investments, wide geographic coverage, fast deployment, and pleasing aesthetics (no unsightly wires).

On the other hand, these technologies are susceptible to scalability issues. For instance, cellular technologies are subject to the spectrum crunch problem, which drives demand for technology optimizations and cellular off-load solutions such as Wi-Fi and femtocell. Also, wireless mesh technologies suffer from challenges such as forwarding latency and slow convergence as the diameter of the mesh scales.

### *5.1.2 Industry Progress*

Now that we have covered the main challenges that IoT presents to the Link layer of the protocol stack, we will shift our focus to describe the industry's progress in addressing those challenges through open standard solutions.

#### 5.1.2.1 IEEE 802.15.4

IEEE 802.15 Task Group 4 (TG4) was chartered to investigate a low data rate wireless connectivity solution with focus on very low complexity and extended battery life span that is in the range of multiple months to multiple years. The solution was meant to operate in an unlicensed, international frequency band. While initial activities of the task group focused on wearable devices, i.e., personal area networks, the eventual applications proved to be more diverse and varied. Potential applications of the solution include sensors, interactive toys, smart badges, remote controls, and home automation. As can be seen from the applications, the focus of the solution has primarily revolved around enabling "specialty," typically short-range, communication.

The resulting IEEE 802.15.4 technology is a simple packet-based radio protocol aimed at very low-cost, battery-operated devices (whose batteries last years) that can intercommunicate and send low-bandwidth data to a centralized device. The protocol supports data rates ranging from 1Mbps to 10 kbps. The data rate is dependent on the operating frequency as well as on the coding and modulation scheme. The standard operates over several frequency bands, which vary by region:

- 169 MHz band
- 450 MHz band
- 470 MHz band
- 780 MHz band
- 863 MHz band
- 896 MHz band
- 901 MHz band
- 915 MHz band
- 917 MHz band
- 920 MHz band
- 928 MHz band
- 1427 MHz band
- 2450 MHz band

In addition, the standard supports multiple modulation schemes, including BPSK, ASK, O-QPSK, MR-FSK, MR-OFDM, and MR-O-QPSK. The transmission range varies from tens of meters up to 1 kilometer, the latter introduced with IEEE 802.15.4g. The protocol is fully acknowledged for transfer reliability. The basic frame size is limited to 127 bytes in the original specification, and the philosophy behind that is twofold: to minimize power consumption and to reduce the probability of frame errors. However, with IEEE 802.15.4g, the maximum frame size is increased to 2047 bytes, accompanied by an increase of the frame check sequence (FCS) from 16 to 32 bits for better error protection.

The standard offers optional fully acknowledged frame delivery for transfer reliability in lossy environments (e.g., high interference). If the originator of a frame does not receive an acknowledgment after a certain time period, it assumes that the transmission failed and retransmits the frame. If an acknowledgment is still not

received after multiple attempts, the originator may either terminate the transaction or continue retrying.

The IEEE 802.15.4 standard only defines the functions of the Physical and Media Access Control (MAC) layers. It serves as the foundation for several protocol stacks, some of which are non-IP, including Zigbee, Zigbee RF4CE, Zigbee Pro, WirelessHART, ISA 100.11a, and RPL.

There are two types of devices in an 802.15.4 network. The first one is the *full-function device* (FFD). It implements all of the functions of the communication stack, which allows it to communicate with any other device in the network. It may also relay messages, in which case it is dubbed as a personal area network (PAN) coordinator. The PAN coordinator is in charge of its network domain: it allocates local addresses and acts as a gateway to other domains or networks. The second type of device is the *reduced-function device* (RFD). RFDs are meant to be extremely simple devices with very modest resource and communication capabilities. Hence, they can only communicate with FFDs and can never act as PAN coordinators. The rationale is that RFDs are to be embedded into the "things." Networks can be built using either a star, mesh, or cluster tree topology (Fig. 5.3). In all three cases, every network needs at least a single FFD to act as the PAN coordinator. Networks are thus formed from clusters of devices separated by suitable distances.

In the star topology, all devices communicate through a single central controller, namely, the PAN coordinator. This is a hub-and-spoke model: the PAN coordinator is the hub, and all other devices form spokes that connect only to the hub. The PAN coordinator is typically main powered, while the devices are most likely battery operated. Use cases that make use of this topology include smart homes (home automation), computer peripherals, personal health monitors, toys, and games. Each star network chooses a PAN identifier, which is not currently in use by any other network within the radio range. This allows each star network to operate independently of other networks.

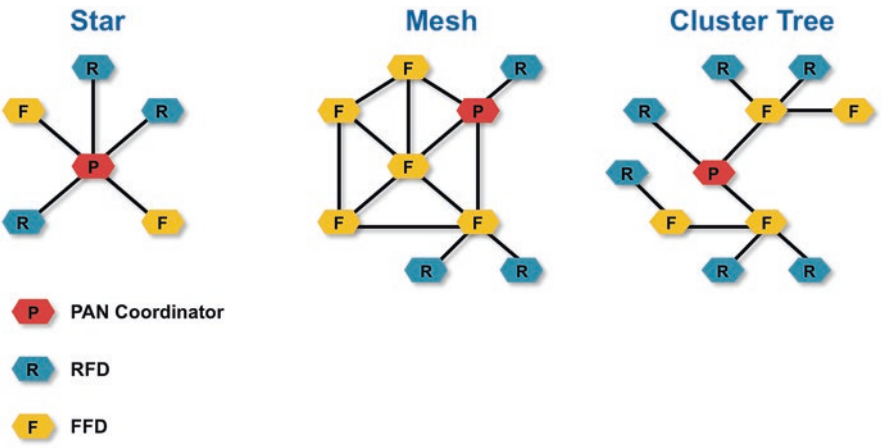



**Fig. 5.3** IEEE 802.15.4 topologies

The mesh topology (also called peer to peer) differs from the star topology in that any device can communicate with any other device as long as the two are within radio range. A mesh network can be ad hoc in formation, self-organizing, and self-healing on node or link failures. It also provides reliability through multipath routing. Use cases such as industrial control and process monitoring, wireless sensor networks (WSN), precision agriculture, security, asset tracking, and inventory management all can leverage this topology.

The cluster tree topology is a special case of a mesh network that comprises of chained clusters. In a cluster tree, the majority of the devices are FFDs. RFDs may connect to the network as leaf nodes at the end of a tree branch. As with any 802.15.4 topology, the network has a single PAN coordinator. The PAN coordinator forms the first cluster by declaring itself as the cluster head (CLH) with a cluster identifier (CID) of zero, selecting an unused PAN identifier, and broadcasting beacon frames to other neighbor devices. A device, which receives beacon frames, may request from the CLH to join the cluster. If the CLH allows the device to join, it will add the new device as a child device in its neighbor list. The newly joined device will add the CLH as its parent in its neighbor list and commence broadcasting periodic beacon frames. This allows other candidate devices to join the same cluster at that device. Once the requirements of the application or network are met, the PAN coordinator may instruct a device to become the CLH of a new cluster that is adjacent to the first. The advantage of this daisy-chained cluster structure is the ability to achieve larger coverage area at the expense of increased message latency.

#### 5.1.2.2 IEEE 802.15.4e TSCH

IEEE 802.15.4e is the next-generation 802.15.4 wireless mesh standard. It aims to improve on its predecessor in two focus areas: lower energy consumption and increased reliability. The standard introduces a new media access control (MAC) layer to 802.15.4 while maintaining the same physical (PHY) layer. Hence, it can be supported on existing 802.15.4 hardware. Two key capabilities are added, time synchronization and channel hopping, hence the acronym TSCH. Time synchronization addresses the requirement for better energy utilization, whereas channel hopping aims at increasing the reliability of communication.

With time synchronization, time is sliced into fixed-length time slots and all nodes are synchronized. A time slot is long enough to allow a station to send a maximum transmission unit (MTU)-sized frame and receive an acknowledgment back. Time slots are grouped into slotframes of flexible width. The flexibility allows different deployments to optimize for bandwidth or for energy saving: the shorter the slotframe, the more frequently that a given time slot will be repeated, thereby giving a station more chances to transmit (i.e., higher bandwidth) but at the expense of increased energy consumption. The current time slot is globally known to all nodes in the network via an absolute slot number (ASN). The ASN is initialized to 0 and is expected to wrap around only after hundreds of years.

With channel hopping, each message transmission between nodes occurs on a specified channel offset. The channel offset is then mapped to a radio frequency using a function that guarantees that two consecutive transmissions between two nodes hop from one frequency to another within the allotted band:

$$\text{Frequency} = F\left\{\left(\text{ASN} + \text{Channel Offset}\right) \text{mod} \text{nFreq}\right\}$$

where nFreq is the number of available frequencies in the allotted band.

This enhances the reliability of communication as it is proven to be effective against multipath fading and interference. Basically, if a specific frequency is subject to fading or interference, then by changing the frequency used for communication between nodes with every new message, only a subset of the messages will be lost due to those conditions, whereas if all communication were to occur on the same frequency, then all messages between the nodes communicating over the affected frequency would be lost during the fading or interference event.

The nodes in the network all obey a TSCH schedule. The schedule is a logical two-dimensional matrix with one dimension determining the slot offset in the slotframe and the second dimension designating the channel offset in the available frequency band (Fig. 5.4). The schedule instructs each node on what it is supposed to do in a given time slot: transmit, receive, or sleep. The schedule also indicates for every communicating node its neighbor's address and the channel offset to be used for said communication. The width of the schedule is equal to the slotframe width, whereas the depth of the schedule is equal to the number of available frequencies in the allotted band. Each cell in the schedule corresponds to a unique slot offset and channel offset combination. The organization of communication in the schedule allows the network to operate using collision-free communication, by ensuring that only a single station transmits in a given cell. Alternatively, it can allow the network to operate in a slotted Aloha paradigm (i.e., carrier-sense multiple access with collision detection—CSMA/CD) by allowing multiple stations to transmit in the same cell. IEEE 802.15.4e does not define the mechanisms by which the TSCH schedule is built and leaves that responsibility to upper-layer protocols.

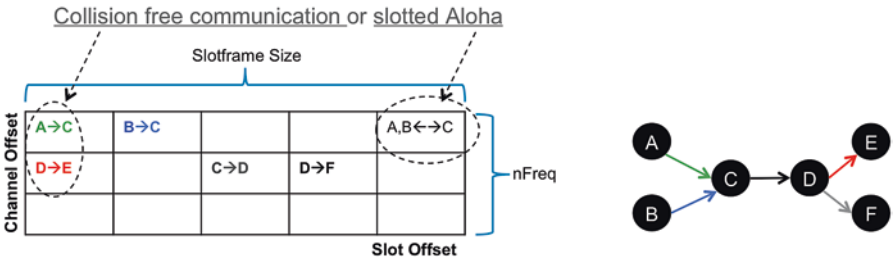



**Fig. 5.4** TSCH schedule

#### 5.1.2.3 LoRaWAN

LoRaWAN defines a communication protocol and network architecture for low-power wide area networks (LPWANs). LoRaWAN is designed to address the requirements for low power consumption (i.e., long battery life), long range, and high capacity in LPWANs while maintaining low cost for the solution. The communication protocol used in LoRaWAN is known as LoRa. The LoRa physical layer uses chirp spread spectrum modulation. It is characterized by low power usage while at the same time significantly increasing the communication range when compared to frequency-shifting keying (FSK), which is the modulation technique often used in legacy wireless systems. Chirp spread spectrum is not a new technique: it has been employed in military and space applications for decades because of its extended range and its robustness against interference. A key advantage of the LoRa protocol is its extended range: a single base station can cover hundreds of square miles. That's enough to provide coverage over cities. Hence, with minimal infrastructure, entire countries can be covered using LoRaWAN. In wireless communication systems, the range within a given environment is determined through the link budget metric. LoRa has a link budget that is greater than any other standardized wireless communication technology today. The link budget is defined as an accounting of all the gains and losses between a transmitter and a receiver:

$$\text{Link Budget} = \text{Transmitted Power} + \text{Gains} - \text{Losses}$$

##### 5.1.2.3.1 LPWAN Motivation

LPWANs are meant to fill the gap between short-range wireless and cellular communication technologies. They are designed for low-power, long-range, and light-weight data collection IoT use cases (Fig. 5.5). Devices connecting to LPWANs will typically have a battery life of over 10 years and will require outdoor coverage of up to 20 km (12 miles) and sufficient indoor penetration. From an operational standpoint, the solutions require low service cost and endpoint complexity. In general, the LPWAN landscape spans both licensed and unlicensed spectrums. LoRaWAN falls under the latter.

##### 5.1.2.3.2 Network Architecture

LoRaWAN employs a long-range star (or hub and spoke) architecture in order to minimize power consumption. Star architecture, in contrast to mesh architecture, eliminates the scenario where nodes receive and forward information from other nodes that is mostly irrelevant to them. In LoRaWAN, gateways act as hub nodes, whereas end devices form the spokes. End nodes are not associated with a particular gateway. Rather, when a node sends data, it is typically received by multiple

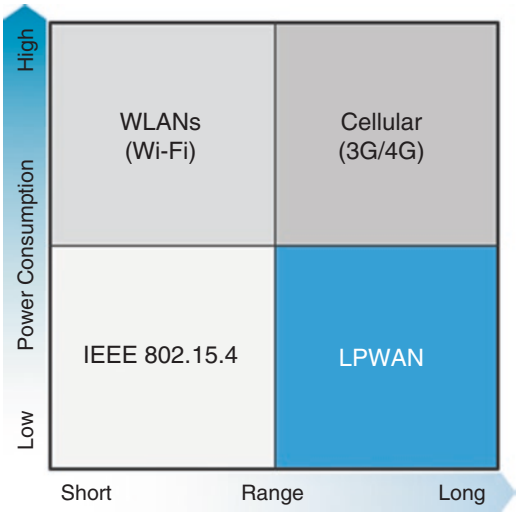



**Fig. 5.5** LPWAN positioning

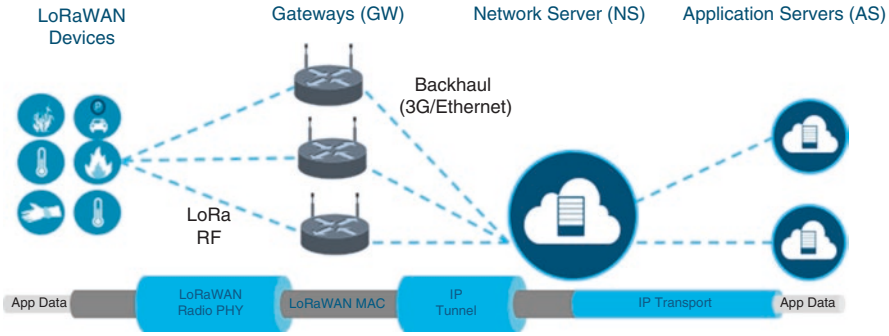



**Fig. 5.6** LoRaWAN end-to-end network architecture

gateways. Each of these gateways, in turn, forwards the received data toward the cloud-based network server using some backhaul[1] technology. The network server is responsible for all complex and intelligent functions: it manages the network, filters redundant received data, performs security verification, schedules acknowledgments through the most optimal gateway, and performs adaptive rate control, etc.

A key feature of this architecture is that no handover mechanism is required from one gateway to another to support the mobility of end nodes. Therefore, it is straightforward to enable IoT asset tracking applications. Another key feature is the built-in access redundancy, where the failure of a gateway or path toward the network server is handled by sending redundant copies of data packets (Fig. 5.6).

[1] The backhaul can be Ethernet, Wi-Fi, satellite, or cellular.

#### 5.1.2.3.3 Device Class Capabilities

In order to address the constrained devices requirement of IoT, LoRaWAN defines three device class capabilities targeting different applications with varying needs. The classes are labeled A, B, and C. They offer a trade-off between energy consumption and downlink communication latency.

Class A devices support bidirectional communication. They include battery-powered sensors. This is the most energy-efficient device class capability and must be supported by all devices implementing LoRaWAN. The communication model is such that each uplink transmission by the end device is followed by two short downlink receive windows. The transmission schedule of the end device is dictated by its own communication requirements, albeit with a small variation in the allocated window based on a random time variance (ALOHA protocol flavor). This class of operation is suitable for applications where downlink communication from the server to the end device mostly occurs in the short window after the latter had sent an uplink transmission. Otherwise, such downlink communication must be deferred till the next scheduled uplink transmission.

Class B devices support bidirectional communication with scheduled receive slots. They include battery-powered actuators. This class offers energy efficiency with latency controlled downlink communication. The communication model for this class supports all the capabilities of Class A and in addition requires end devices to open extra receive windows at scheduled times. This is accomplished by having the end devices receive a time-synchronized beacon from the gateways, so that the applications on the servers know when the end devices are listening on these extra slots.

Class C devices support bidirectional communication with maximal receive slots. They include main powered actuators. This class is for devices that have the energy resources to afford to listen continuously. It is well suited for applications that require no latency in downlink communication. End devices in this class must continuously open receive windows, when not in transmitting mode.

#### 5.1.2.3.4 Scalability

LoRaWAN ensures the scalability of its long-range star network architecture through high-capacity gateways. Gateways achieve high capacity through a twofold approach, by using adaptive data rate and by employing a multichannel multi-modem transceiver. This allows the gateway to receive simultaneous messages on multiple channels from a very high volume of end devices. Several factors affect network capacity, among which the following are deemed most critical:

- Number of concurrent channels supported by the transceiver
- Data rate (i.e., time on air)
- Payload size
- Frequency of transmission of communicating nodes

Recall that LoRa uses spread spectrum modulation; hence, when different spreading factors are used, the signals end up being orthogonal to one another. The effective data rate changes with change in the spreading factor. LoRaWAN gateways capitalize on this property in order to concurrently receive multiple different data rates on the same channel. In the scenario where an end device is in the vicinity of a gateway and has a good link, there is no technical reason for it to use the lowest data rate thereby filling up the available spectrum for a longer time period than required. If this device were to shift to a higher data rate, its time on air will be shortened, thereby freeing up more time for other devices to transmit. It is worth noting that in order for adaptive data rate to work, the uplink and downlink need to be symmetrical, with sufficient downlink capacity. These features all contribute to making a LoRaWAN network scalable.

However, the duty-cycle limitation in the ISM bands may arise as a limitation to the scale of LoRaWAN networks. As an example, the maximum duty cycle of the EU 868 ISM band is 1%. This results in a maximum transmission time of 36 s in each hour for each end device in a sub-band.

#### 5.1.2.3.5 Energy Efficiency

Energy efficiency is achieved in LoRaWAN through the use of the ALOHA method of communication: nodes are asynchronous and only communicate when they have data ready to be sent, whether scheduled or event driven. This alleviates the need for end devices to frequently wake up and synchronize with the network or check for messages. Such synchronization is one of the primary contributors to energy consumption in wireless networks.

Energy efficiency is also achieved through the use of adaptive data rate, where transmission power is varied according to link quality. When adaptive data rate is enabled, the network collects metrics on a number of the most recent transmissions from a node. These metrics include the frame counter, signal-to-noise ratio (SNR), and the number of gateways that have received each transmission. Based on these metrics, the network then calculates if it is possible to increase the data rate or lower the transmission power. If possible, the network will lower the transmission power to save energy and cause less interference.

#### 5.1.2.3.6 Security

LoRaWAN defines two layers of security: one at the Network layer and one at the Application layer. Network security is responsible for ensuring the authenticity of the node in the network, whereas the Application layer security guarantees that the user's application data is inaccessible to the network operator. LoRaWAN uses AES encryption with key exchanges based on the IEEE EUI64 identifier.

Three different security keys are defined: network session key, application session key, and application key. The network session key is used for securing the

interactions between the end node and the network. It helps in checking the validity of the messages. The application session key is used for payload encryption/decryption. These two session keys are unique per device, per session. When a device is dynamically activated, these keys are regenerated upon every activation, whereas, if the device is statically activated, these keys remain the same until changed by the operator. Devices which are dynamically activated use the application key in order to derive the two session keys in the course of the activation procedure. In general, it is possible to have either a default application key that is used to activate all devices or a customized key per device.

##### 5.1.2.3.7 Regional Variations

Due to differences in spectrum allocations and regulatory requirements between regions, the LoRaWAN specification varies slightly from region to region. These variations affect the following: frequency band, number of channels, channel bandwidth, transmission power, data rate, link budget, and spreading factor.

##### 5.1.2.3.8 Challenges

LoRaWAN relies on the acknowledgment of frames in the downlink for reliability. This, in turn, causes capacity drain. Therefore, in general, application should try to minimize the volume of acknowledgments in order to avoid this drain. This raises an open question regarding the feasibility of very large-scale and ultrareliable applications using LoRaWAN.

Also, the uncoordinated deployment of LoRaWAN gateways and alternate LPWAN technologies in large urban centers may lead to a decrease in network capacity due to collisions in the ISM bands. This, in addition to the duty-cycle regulation for these bands, poses potential challenges for large-scale LoRaWAN deployments.

### 5.1.2.4 IEEE 802.11ah

The popularity of IEEE 802.11 wireless technologies (Wi-Fi) has grown steadily over the years in home, business, as well as metropolitan area networks. The technology, however, cannot sufficiently address the requirements of IoT, due to the following two reasons:

- High power consumption for client stations: Wi-Fi has the reputation of not being very power efficient, due to the need for client devices to wake up at regular intervals to listen to AP announcements, waste cycle in contention processes, etc.

- Unsuitable frequency bands: Wi-Fi currently uses the 2.4–5 GHz frequency bands, which are characterized by short transmission range and high degree of loss due to obstructions. A common solution to this is the use of repeaters, but those add to the power consumption of the solution and add to the network's complexity.

To address these issues, IEEE 802.11 formed Task Group "ah." The 802.11ah group was chartered to develop a wireless connectivity solution that operates in the license-exempt sub-1GHz bands to address the following IoT requirements: large number of constrained devices, long transmission range, small (approximately 100 bytes) and infrequent data messages (inter-arrival time larger than 30 s), low data rates, and one-hop network topologies. The solution is intended to provide a transmission range of up to 1 km in outdoor areas with data rates above 100 kbps while maintaining the current Wi-Fi experience for fixed, outdoor, point-to-multipoint applications. From a design philosophy perspective, the solution optimizes for lower power consumption and extended range at the expense of throughput, where applicable. In addition, the solution aims for scalability by supporting a large number of devices (up to 8191) per Wi-Fi access point.

IEEE 802.11ah introduces new PHY and MAC layers. The new layers are designed for scalability, extended range, and power efficiency. Compared to existing Wi-Fi technologies which operate in the 2.4–5 GHz range, the use of the sub-1GHz band provides longer range through improved propagation and allows better penetration of the radio waves through obstructions (e.g., walls).

However, one of the challenges in the use of the sub-1GHz spectrum is that its availability differs from one country to the next, with large channels available in the United States, whereas many other regions only have a few channels. This led the 802.11ah group to create several channel sizes: 1, 2, 4, 8, and 16 MHz channels based on the needs and regulatory domains of different countries. It also led the group to define operation over several frequency bands, which vary by region:

- Europe: 868–868.6 MHz.
- Japan: 950–958 MHz
- China: 314–316, 390–434, 470–510, and 779–787 MHz
- Korea: 917–923.5 MHz
- United States: 902–928 MHz

IEEE 802.11ah will support data rates ranging from 150 kbps up to 340 Mbps. The supported modulation schemes include BPSK, QPSK, and 16 to 256 QAM.

In order to address the IoT requirements of low-power consumption and massive scalability, the emerging 802.11ah introduces several enhancements to Wi-Fi technology that can be categorized into three functional areas:

- Providing mechanisms for client stations to save power through longer sleep times and reducing the need to wake up.
- Improving the mechanisms by which a client station accesses the medium by providing procedures to allow the station to know when it will be able to, or will have to, access the channel.

- Enhancing the throughput of a client station that accesses the channel, by reducing the overhead associated with current IEEE 802.11 exchanges through reducing frame headers, as well as simplifying and speeding management frames exchanges.

In what follows, we will describe a number of those enhancements in more detail.

#### 5.1.2.4.1 Short MAC Header

To enhance throughput, 802.11ah adds support for a shorter MAC header compared to the current 802.11 standard. Information contained in the QoS and HT control fields (the latter introduced to the MAC header with 802.11n) are moved to a signal (SIG) field in the PHY header. The other non-applicable parts of the header are suppressed, e.g., no duration/ID fields, since there is no virtual clear channel assessment (CCA). The new header is 12 bytes shorter than the standard 802.11n header. Following the same logic, the acknowledgment (ACK) frame is replaced with a null data packet, which only contains the PHY header (no MAC header, no FCS). That frame is sent at a special reserved modulation and coding scheme (MCS) to make it recognizable. MCS is a simple integer assigned to every permutation of modulation, coding rate, guard interval, channel width, and number of spatial streams.

#### 5.1.2.4.2 Large Number of Stations

To enable support for a large number of client stations, 802.11ah extends the Association Identifier (AID), which is limited to 2007 in the current 802.11 standard, by creating a hierarchical identifier with a virtual map, bringing the number up to 8191.

#### 5.1.2.4.3 Speeding Frame Exchanges

In current 802.11 frame exchanges, a client station first has to contend for the medium, then transmit its frames, and then wait for an acknowledgment from the access point (AP). If the client station expects a response, it has to stay awake, while the AP contends for the medium and then sends. The client station finally sends an acknowledgment. With the 802.11ah speed frame exchange mechanism, the dialog can occur within a single transmission opportunity (TXOP): the client station wakes up, contends for the medium, and sends the frame to the AP, and the AP immediately replies after just a short inter-frame gap, allowing the client station (e.g., sensor) to immediately go back to sleep mode after receiving the answer, saving on uptime wasted in inter-frame and two-way acknowledgments.

#### 5.1.2.4.4 Relay

Client stations often need to exchange information with one another, going through one or more intermediary APs when a direct connection is not available. In such exchanges, the client stations are forced to stay awake for the entire duration of the dialog. This process is greatly optimized with 802.11ah relay coupled with speed frame exchange. The client station wakes up and sends a frame to the AP, asking the latter to relay. The client station can then immediately go back to sleep/power-saving mode. The AP may relay the frame through another AP or deliver it directly to the destination. This model is appealing due to a number of reasons: the AP is usually main powered and has enough resources to buffer the frame until the destination client station wakes up. The same process can be repeated for the response message, allowing both client stations to optimize power consumption when they are not actively sending or receiving. This also eliminates the need for the client stations to synchronize wake/sleep cycles.

#### 5.1.2.4.5 Target Wake Time

With target wake time (TWT), the AP can inform client stations when they will gain the right to access the medium. A client station and an AP can exchange initial frames expressing how much access the former needs. Then, the AP can assign a target wake time for the station, which can be either aperiodic or periodic (thus eliminating the need for the client station to have to wake up to listen to TWT values). Outside of the TWT, the client station can sleep and does not have to wake up to listen to any messages, not even beacon frames. At those target wake times (TWTs), the AP can send a null data packet paging (NDP) that tells the client station about the AP buffer status. This allows the AP to smoothly deliver buffer content to all client stations one after the other, instead of having all stations wake up at beacon time.

#### 5.1.2.4.6 Grouping

Client stations can be grouped based on their location, using a group identifier assignment that relies on their type or other criteria. The AP then announces which groups are allowed to be awake for the next time period and which groups can go back to sleep mode because they will not be allowed to access the channel. This saves battery power on the sleeping groups, as these do not have to listen to the traffic. This logic brings a form of time division multiplexing (TDM) to Wi-Fi, by allowing transmission to each group based on time periods.

#### 5.1.2.4.7 Traffic Indication Map (TIM) and Paging Mechanism

802.11ah introduces a traffic indication map (TIM) and page segmentation mechanism, by which an AP splits the TIM virtual bitmap into segments and each beacon only carries one segment. This allows IoT devices to wake up only to listen to the TIM matching their segment number. 802.11ah also introduces the concept of TIM stations (that need to get TIM info and therefore wake up at regular intervals) and non-TIM stations (that do not expect to receive anything and therefore can sleep beyond TIMs and do not need to wake up unless they need to send).

#### 5.1.2.4.8 Restricted Access Windows

The AP can define a restricted access window (RAW), which is a time duration composed of several time slots. The AP can inform client stations that they have the right to send or receive only during certain time slots within the window, in order to distribute traffic evenly. The AP would use the RAW parameter set (RPW) to determine and communicate these slots and transmission or reception privileges. A client station that has traffic to send upstream but for which the AP does not have traffic to send downstream can send a request message to indicate to the AP that it needs a slot upstream.

### 5.1.2.5 Comparison of Wireless Link Layer Protocols

The table below summarizes key characteristics of the wireless IoT link layer protocols discussed in this chapter:

| Protocol | Range | Data rate | Topology | Application | Power consumption |
|---|---|---|---|---|---|
| IEEE 802.15.4 | Up to 1 km | 1Mbps to 10Kbps | Mesh | Personal area network/home network | Very low |
| LoRaWAN | Up to 20 km | Upto 50Kbps | Star | Wide area network | Low |
| IEEE 802.11ah | Up to 1 km | >100Kbps | Star | Metropolitan block | Medium |

### 5.1.2.6 Time-Sensitive Networking

The requirements for Time-Sensitive Networking originate from real-time control applications such as industrial automation and automotive networks. These requirements contribute to some of the most prominent gaps that current Internet

technologies need to address at the Link layer to realize the vision of IoT. In the case of industrial automation, the networks are relatively large (in the order of one to several kilometers) and may include up to 64 hops for a factory and up to 5 hops within a work cell (e.g., robot). The network needs to accommodate, in addition to real-time control traffic, other long-tailed traffic such as video or large file transfers. One of the key requirements for such networks is precise time synchronization, in the order of ± 500 nanoseconds within a work cell and ± 100 microseconds factory wide. Another key requirement is deterministic delay, which is not to exceed 5 microseconds within a work cell and 125 microseconds factory wide. Last but not least, a fundamental requirement for such networks is high availability as it is critical for the safety of the operators. This translates to a requirement for redundant paths with seamless or instantaneous switchover time, not to exceed 1 microsecond. In the case of automotive networks, the physical size of the deployments is relatively small, but the number of ports required is large: as an example, the network may span 30 meters over 5 hops with over 100 devices connected (sensors, radar, control, driver-assist video, information, and entertainment audio/video). A key requirement for these networks is support for deterministic and very small latency, less than 100 microseconds over 5 hops using 100Mbps links. Another important requirement is high availability to ensure driver and passenger safety.

The above networks have typically been based on non-IP technologies. Connectivity has traditionally been achieved using some fieldbus technology such as DeviceNet, Profibus, and Modbus. Each of these technologies conforms to specific power, cable, and communication specifications, depending on the supported application. This has led to the situation where multiple desperate networks are deployed in the same space and has driven the need to have multiple sets of replacement parts, skills, and support programs within the same organization. With IoT, it will be possible to unite these separate networks into a converged network infrastructure based on industry standards. A candidate set of technologies to provide the Link layer functions of this converged network infrastructure is the IEEE 802 family of local area network (LAN)/metropolitan area network (MAN) protocols. One of the more popular technologies in the IEEE 802 family of protocols is Ethernet. Ethernet is by far the most widely deployed LAN technology today, connecting more than 85 percent of the world's local area networks (LANs). More than 300 million switched Ethernet ports have been installed worldwide. Ethernet's ubiquity can be attributed to the technology's simplicity, plug-and-play characteristics, and ease of manageability. Furthermore, it is low cost and flexible and can be deployed in any topology. Ethernet and the IEEE 802 family of protocols have steadily evolved over the years, with the IEEE Audio-Video Bridging (AVB) task group focusing on standards for transporting latency-sensitive traffic over bridged networks, primarily for multimedia (audio and video) streaming applications. These standards provide a foundation on which to build Time-Sensitive Networking technologies for IoT. They provide architecture for managing different classes of time-sensitive traffic through a set of in-band protocols. In particular, IEEE 802.1AS defines a profile for the Precision Timing Protocol (PTP), which provides time synchronization of distributed end systems over the network with accuracy better than

± 1 microseconds. IEEE 802.1Qav defines forwarding and queuing rules for time-sensitive traffic in Ethernet. It specifies two traffic classes, class A and class B, with maximum latency guarantees of 2 milliseconds and 50 milliseconds, respectively. Traffic that does not belong to one of these two classes is considered to be "best effort," which includes all legacy Ethernet traffic. Traffic shaping and transmission selection are performed using a credit-based shaping algorithm: traffic is organized by priority, according to its class, and transmission of a frame in one of the above two classes is only allowed when credits are available for the associated class. Upper and lower bounds on the credit-based shaper limit the bandwidth and burstiness of the streams. Furthermore, IEEE standard 802.1Qat (part of IEEE 802.1Q-2011) defines a signaling protocol for dynamic registration and resource reservation of new streams, which provides per-hop delays in the order of 130 microseconds on 1 Gbps Ethernet links.

These standards, however, fall short in a number of areas: First, IEEE 802.1AS can take up to 1 s to switch to a new grandmaster clock (GMC) in the case of failure of the primary GMC. For real-time control applications, it is required to have the switchover time be in the order of 250 milliseconds or less. Also, it is highly desirable to support multiple concurrently active GMCs for high availability. Second, per-hop switch delays need to be reduced by almost two orders of magnitude. Third, path selection and reservation for critical streams need to be made faster and simpler in order to accommodate high-scale deployments with thousands of streams.

As discussed previously, network high availability is of paramount importance in real-time IoT applications. Ethernet has historically, and for a long period of time, relied on the Spanning Tree Protocol (STP) in order to support redundancy and failure protection. However, in the past decade or so, requirements for massively scalable Ethernet networks in data center and metropolitan area network (MAN) deployments have resulted in the evolution of the Ethernet plane toward the use of the Intermediate System-to-Intermediate System (IS-IS) protocol, as defined in IEEE 802.1aq-2012 (Shortest Path Bridging) and IEEE 802.1Qbp-2014 (Equal Cost Multiple Path). IS-IS provides mechanisms for topology discovery and setup of redundant paths. It also includes mechanisms for network reconfiguration in the case of failures with reasonable delays (better than STP). These standards, however, are still lacking in the following areas: There are no standardized mechanisms to engineer paths with nonoverlapping or minimally overlapping links and nodes. Also, there are no mechanisms that provide extremely fast (i.e., instantaneous) switchover in the case of failures. Finally, there are no mechanisms for redundant (simultaneous) transmission of streams along nonoverlapping paths.

The IEEE Time-Sensitive Networking TSN task group was formed in November 2012, by renaming the Audio/Video Bridging (AVB) task group, with the goal of addressing the gaps highlighted above. Under that umbrella, work on three emerging standards commenced: 802.1Qca Path Control and Reservation, 802.1Qbv Enhancements for Scheduled Traffic, and 802.1CB.

#### 5.1.2.6.1 IEEE 802.1Qca

This emerging standard extends the use of IS-IS to control Ethernet networks beyond what is defined in IEEE 802.1aq Shortest Path Bridging. It provides explicit path control, bandwidth, and stream reservation and redundancy (through protection or restoration) for data streams. It proposes the use of IS-IS for topology discovery and to carry control information for scheduling and time synchronization. The new protocol will enable the use of non-shortest paths and will provide explicit forwarding path (explicit tree—ET) control. Path calculation and determination will be done through a Path Computation Element (PCE), the latter being defined by the IETF PCE workgroup. The PCE is an application that computes paths between nodes in the network based on a representation of its topology. In 802.1Qca, IS-IS is currently being proposed as the protocol to convey the topology information from the Ethernet network to the PCE. The PCE may be centralized and reside in a dedicated server or in a network management system (NMS), or it may be distributed and embedded in the network elements (e.g., routers or bridges) themselves.

The diagram below shows an example Ethernet network controlled by a single PCE residing in end station X. This end station is connected to SPT Bridge 11. The PCE peers with the bridge using IS-IS to learn the topology. The PCE can compute explicit trees based on, for example, bandwidth or delay requirements, and communicates them using IS-IS extensions to the bridges (Fig. 5.7).

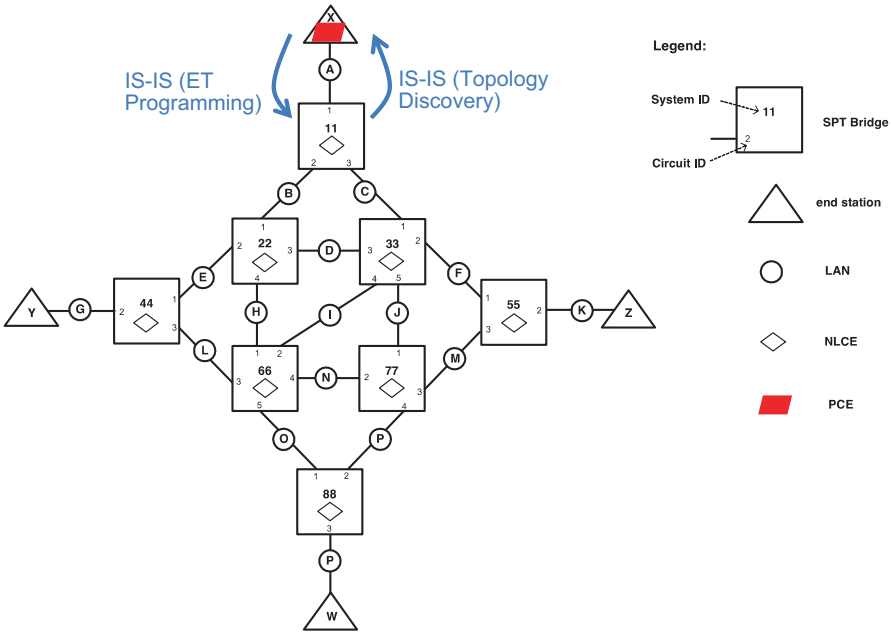



**Fig. 5.7** Example IEEE 802.1Qca network

#### 5.1.2.6.2 IEEE 802.1Qbv

The IEEE 802.1Qbv standard will provide real-time control applications with performance assurances for network delay and jitter over "engineered" LANs while maintaining coexistence with IEEE 802.1Qav/Qat reserved streams and best-effort traffic on the same physical network. Engineered LANs are so called because traffic transmission schedules for the network can be designed offline. These pre-configured schedules assign dedicated transmission slots to each node in the network, for the purpose of preventing congestion and enabling isochronous communication with deterministic latency and jitter. The emerging standard will define time-aware shaping algorithm that enables communicating nodes to schedule the transmission of messages based on a synchronized time. It is proposed that priority markings carried in the frames will be used to distinguish between time-scheduled, reserved stream (credit based), and best-effort traffic.

The figure below depicts the traffic queue architecture for a bridge port that implements this emerging standard. A transmission gate is associated with each traffic queue; the state of the transmission gate determines whether or not queued packets can be selected for transmission on the port. Global Gate Control logic determines what set of gates are open or closed at any given point of time. A packet on a queue cannot be transmitted if the transmission gate, for that queue, is in the closed state or if the packet size is known and there is insufficient time available to transmit the entirety of that packet before the next gate-close event associated with that queue (Fig. 5.8).

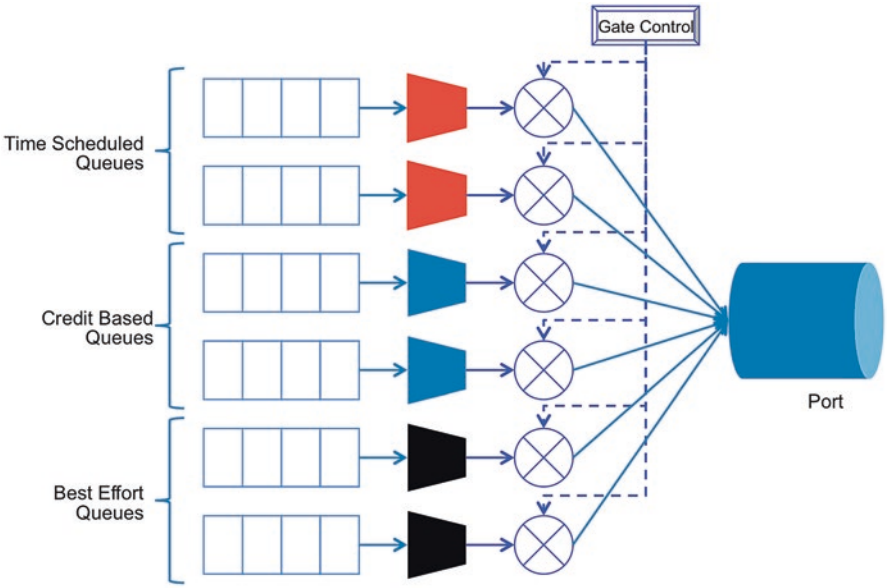



**Fig. 5.8** IEEE 802.1Qbv time-based queuing

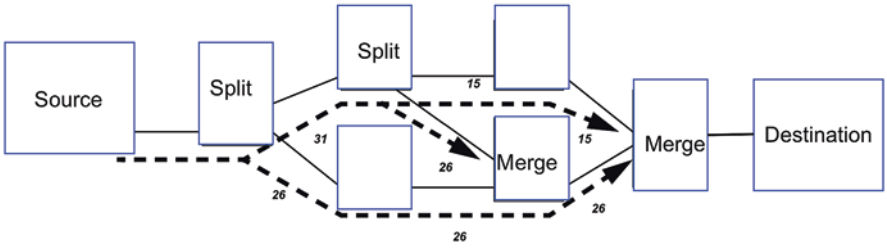



**Fig. 5.9** IEEE 802.1CB seamless redundancy

#### 5.1.2.6.3 IEEE 802.1CB

In order to maximize the availability and reliability of the network, IEEE 802.1CB proposes mechanisms that will enable "seamless redundancy" over 802.1Qca networks. With seamless redundancy, the probability of packet loss is reduced by sending multiple copies of every packet of a stream. Each copy is transmitted along one of a multitude of redundant paths. Duplicate copies are then eliminated to reconstitute the original stream before it reaches its intended destination.

This is effectively done by tagging packets with sequence numbers to identify and eliminate the duplicates and by defining new functions for bridges, a split function, responsible for replicating packets in a stream, and a merge function responsible for eliminating duplicate packets of a stream (Fig. 5.9).

IEEE 802.1CB proposes introducing a new tag to the 802.1Q frame, the redundancy tag, which includes a 16-bit sequence number. The emerging standard recognizes that alternate tagging mechanisms are possible, for example, through the use of multiple protocol label switching (MPLS) pseudowires [RFC4448] or using IEEE 802.1AE MacSec.

## 5.2 Internet Layer

### *5.2.1 Challenges*

Many IoT deployments constitute what is referred to as low-power and lossy networks (LLNs). These networks comprise of a large number (several thousand) of constrained embedded devices with limited power, memory, and processing resources. They are interconnected using a variety of Link layer technologies, such as IEEE 802.15.4, Bluetooth, Wi-Fi, or power-line communication (PLC) links. There is a wide scope of use cases for LLNs, including industrial monitoring, building automation (HVAC, lighting, access control, fire), connected homes, healthcare, environmental monitoring, urban sensor networks (e.g., smart grid), and asset tracking. LLNs present the following five challenges to the Internet layer of the protocol stack:

Nodes in LLNs operate with a hard, very small bound on state. As such, Internet layer protocols need to minimize the amount of state that needs to be kept per node for routing or topology maintenance functions. The design of LLN routing protocols needs to pay close attention to trading off efficiency for generality, as most LLN nodes do not have resources to spare.

Typically, LLNs are optimized for saving energy. Various techniques are used to that effect, including employing extended sleep cycles, where the embedded devices only wake up and connect to the network when they have data to send. Thus routing protocols need to adapt to operate under constant topological changes due to sleep/wake cycles.

Traffic patterns within LLNs include point-to-point, point-to-multipoint, and multipoint-to-point flows. As such, unicast and multicast considerations should be taken into account when designing protocols for this layer.

LLNs will typically be employed over Link layer technologies characterized with restricted frame sizes; thus routing protocols for LLNs should be adapted specifically for those Link layers.

Links within LLNs may be inherently unreliable with time-varying loss characteristics. The protocols need to offer high reliability under those characteristics.

Internet layer protocols in LLN have to take the above issues and challenges as design requirements. The protocol design should take into account the link speeds and the device capabilities. For example, if the devices are battery powered, then protocols that require frequent communication will deplete the nodes' energy faster. As described above, LLNs are inherently lossy: a characteristic that is typically unpredictable and predominantly transient in nature. The design of the Internet layer protocols must account for these characteristics. In conventional networks, these protocols react to loss of connectivity by quickly reconverging over alternate routing paths. This is to minimize the extent of data loss by routing around link, node, or other failures as quickly as possible (e.g., MPLS fast reroute mechanism strives for reconvergence within 50 milliseconds). In LLNs, such quick reaction to failures is undesirable due to the transient nature of loss in these networks. As a matter of fact, it would lead to instability and unacceptable control plane churn. Instead, the protocols should follow a paradigm of underreacting to failures in order to dampen the effect of transient connectivity loss, combined with confidence-monitoring model to determine when to trigger full reconvergence. The varying link quality levels in LLNs have direct bearing on protocol design, especially with regard to convergence characteristics and time. In traditional networks, global reconvergence is triggered to minimize the convergence time, whereas in LLNs local reconvergence is preferred, where the traffic is locally redirected to an alternate next hop during transient instabilities. This is to minimize the effect of routing instabilities that may lead to overall network oscillations or forwarding loops. Another consideration for LLNs is the dynamic nature of link and node metrics used in route computation. There are so many dynamic factors in LLNs, such as link quality deteriorating due to interference, node switching from mains power to battery power, momentary CPU overload on a node, etc. These factors cause node and link

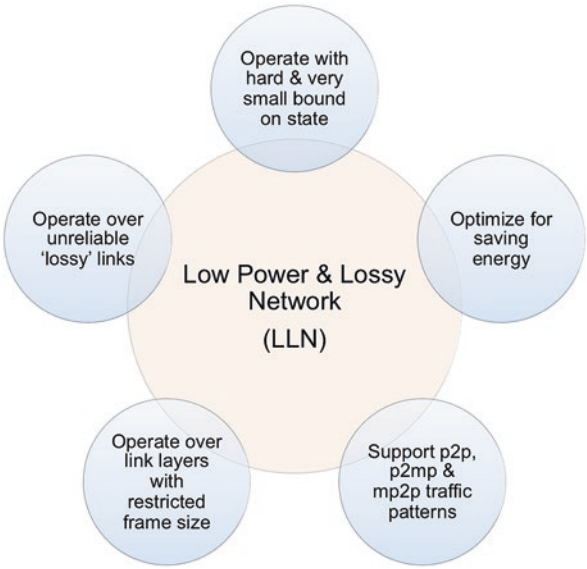



**Fig. 5.10** IoT challenges for the Internet layer. (Source Cisco BRKIOT-2020, 2015)

metrics to be time varying in nature, and the routing protocols must be able to handle that.

Existing routing protocols such as OSPF, IS-IS, etc. in their current form do not satisfy the routing requirements imposed by the above challenges (Fig. 5.10).

## *5.2.2 Industry Progress*

### 5.2.2.1 6LowPAN

As discussed previously, one of the challenges imposed by IoT on the Internet layer is the adaptation of this layer's functions to Link layer technologies with restricted frame size. A case in point is adapting IP, and specifically the scalable IPv6, to the IEEE 802.15.4 Link layer. The base maximum frame size for 802.15.4 is 127 bytes, out of which 25 bytes need to be reserved for the frame header and another 21 bytes for link layer security. This leaves, in the worst case, 81 bytes per frame to cram the IPv6 packet into. What add to the problem are two issues: first, the IPv6 packet header, on its own, is 40 bytes in length, and second, IPv6 does not perform segmentation and reassembly of packets; this function is left to the end stations or to lower layer protocols. Even though 802.15.4 g increases the maximum frame size to 2047 bytes, it is still highly desirable to be able to compress IPv6 packet headers over this Link layer. For the aforementioned reasons, the IETF defined IPv6 over low-power wireless personal area networks (6LowPAN). 6LowPAN is defined in RFC6282. It is an adaptation layer for running IPv6 over 802.15.4 networks (Fig. 5.11). 6LowPAN provides three main functions: IPv6 header compression, IPv6 packet

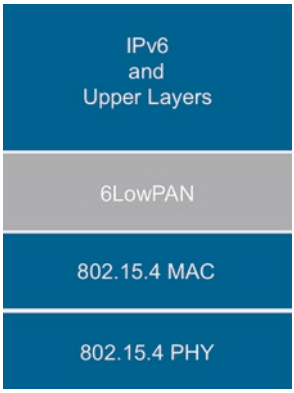



**Fig. 5.11** 6LowPAN Adaptation layer

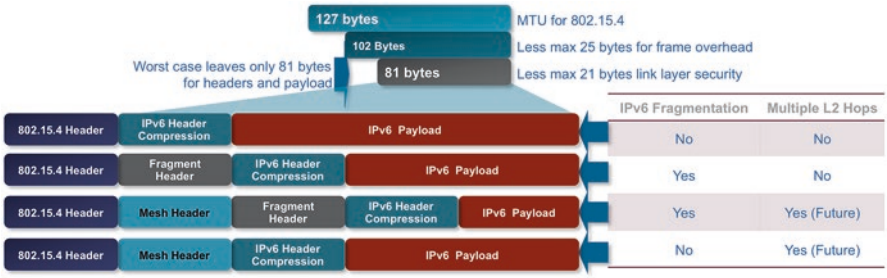



**Fig. 5.12** 6LowPAN header stack. (Source: Cisco BRKIOT-2020, 2015)

segmentation and reassembly, and layer 2 forwarding (also referred to as mesh under). With 6LowPAN, it is possible to compress the IPv6 header into 2 bytes, as most of the information is already encoded into the Link layer header.

6LowPAN introduces three headers for each of the three functions that it provides. Those headers are compression header, fragment header, and mesh header. One or more of these headers may be available in any given packet depending on which functions are applied (Fig. 5.12).

6LowPAN defines new mechanisms to perform IPv6 neighbor discovery (ND) operations including link-layer address resolution and duplicate address detection.

A recurring issue when adapting IPv6 to any Link Layer technology is support for a single broadcast domain, where a host can reach any number of hosts within the subnet by sending a single IP datagram. Accommodating a single broadcast domain within a 6LoWPAN network requires Link layer routing and forwarding functions, often referred to as mesh under, since the multi-hop mesh topology is abstracted away from the IP layer to appear as a single network segment. However, the IETF has not specified a mesh-under routing protocol for 6LoWPAN. Hence, this constitutes a technology gap, especially for IoT applications that can benefit from or that rely on intra-subnet broadcast capabilities.

Even though the scope of 6LoWPAN was originally focused on the IEEE 802.15.4 Link layer, the technology has very limited dependency on 802.15.4 specifics, thereby allowing other link technologies (e.g., power-line communication—PLC) to utilize the same adaptation mechanisms. Consequently, the term "6LoWPAN networks" is often generalized to refer to any Link-layer mesh network built on low-power and lossy links leveraging 6LoWPAN mechanisms.

#### 5.2.2.2 RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks

The routing over low-power and lossy networks (ROLL) workgroup in IETF has defined in RFC 6550 an IPv6 routing protocol for LLNs, known as RPL.[2] RPL is a distance-vector routing protocol. The reason for choosing a distance-vector protocol, as opposed to a link-state paradigm, is primarily to address the requirement of minimizing the amount of control-plane state (memory) that needs to be maintained on the constrained nodes of LLNs. Link-state routing protocols build and maintain a link-state database of the entire network on every node and hence tend to be heavier on memory utilization compared to distance-vector algorithms. RPL computes a destination-oriented directed acyclic graph (DODAG) based on an objective function and a set of metrics and constraints. In the context of routing, a directed acyclic graph (DAG) is formed by a series of nodes and links. Each link connects one node to another in a directed fashion such that it is not possible to start at a node N and follow a directed path that cycles back to node N. A destination-oriented DAG is a DAG that includes a single root node. The DODAG is a logical topology built over the physical network for the purpose of meeting specific criteria and carrying traffic subject to certain requirements. These criteria and requirements are captured in the objective function, metrics, and constraints. The objective function captures the goal behind setting up a specific topology. Example objective functions include minimizing latency of communication or maximizing the probability of message delivery. Metrics are scalar values that serve as input parameters to the best-path selection algorithm. Example metrics include link latency or link reliability or node energy level. Constraints refer to conditions that would exclude specific nodes or links from the topology if they do not meet those constraints, such as exclude battery-powered nodes or avoid unencrypted links. RPL supports dynamic metrics and constraints, where the values change over time and the protocol reacts to those changes.

In a RPL network, a given node may be a member of different logical topologies, or DODAGs, each with a different objective. This is supported through the notion of RPL "instances." An RPL instance is a set of DODAGs rooted at different nodes, all sharing the same objective function (Fig. 5.13).

The DODAG root is typically a border router that connects the LLN to a backbone network. It is always assigned a rank of 1. RPL calculates ranks for all nodes connected to the root based on the objective function. The rank value increases

[2] Pronounced as "ripple"

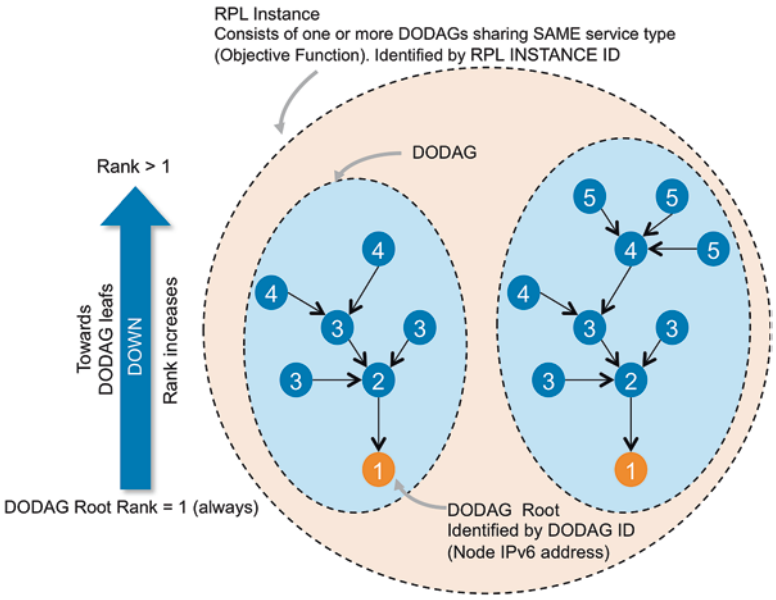



**Fig. 5.13** RPL instances and DODAGs

moving down from the root toward leaf nodes. The rank indicates the node's position or coordinates in the graph hierarchy.

RPL has two characteristics that render it well suited for LLNs: First, it is a proactive protocol, i.e., it can calculate alternate paths as part of the topology setup, as opposed to reactive protocols which rely on exchanging control plane messages after a failure occurs to determine backup paths. Second, RPL is underreactive: it prefers local repair to global reconvergence. Failures are handled by locally choosing an alternate path, which makes the protocol well suited for operation over lossy links.

#### 5.2.2.3 6TiSCH

As discussed previously, IEEE 802.15.4 TSCH defines the medium access control functions for low-power wireless networks with time scheduling and channel hopping. TSCH can fit as the Link layer technology in an IPv6-enabled protocol stack for LLNs, with 6LoWPAN and RPL. The functional gap in the solution is a set of entities that can take control of defining the policies to build and maintain the TSCH schedule, matching that schedule to the multi-hop paths maintained by the RPL routing protocol and adapting the resources allocated between adjacent nodes to traffic flows.

As such, an adaptation layer is required in order to run the IPv6 stack on top of IEEE 802.15.4 TSCH. The IETF has recently formed the 6TiSCH workgroup in order to address this technology gap and define what is referred to as the "6top" adaptation layer. This adaptation layer is sandwiched in between the 802.15.4 link layer and the 6LoWPAN adaptation layer. Its goals are to address the following issues:

#### 5.2.2.3.1 Network Formation

The adaptation layer must control the formation of the network. This includes two functions: the mechanisms by which new nodes securely join the network and the mechanisms by which nodes that are already part of the network advertise its presence.

#### 5.2.2.3.2 Network Maintenance

After the network is formed, the adaptation layer needs to maintain the network's health and ensure that the nodes stay synchronized. This is because a TSCH node must have a time-source neighbor to which it can synchronize at all times. The adaptation layer is responsible for assigning those neighbors to the nodes, to guarantee the correct operation of the network.

#### 5.2.2.3.3 Topology and Schedule Mapping

The adaptation layer needs to gather basic topological information, including node and link state, and provide this information to RPL, so the latter can compute multi-hop routes. Conversely, the adaptation layer needs to ensure that the TSCH schedule contains cells corresponding to the multi-hop routes calculated by RPL.

#### 5.2.2.3.4 Resource Management

The adaptation layer is responsible for providing mechanisms by which neighboring nodes can exchange information regarding their schedule and negotiate the addition or deletion of cells. Note that a cell maps to a transmission/reception opportunity, and, hence, constitutes an atomic unit of resource in TSCH. The number of cells to be assigned between two neighbor nodes should be sized proportionately to the volume of traffic between them.

#### 5.2.2.3.5 Flow Control

While TSCH defines mechanisms by which a node can signal to its neighbors when it can no longer accept incoming packets, it does not, however, specify the policies that govern when to trigger those mechanisms. Hence, it is the responsibility of the adaptation layer to specify mechanisms for input and output packet queuing policies, manage the associated packet queues, and indicate to TSCH when to stop accepting incoming packets. The adaptation layer should also handle transmission failures, in the scenario where TSCH has attempted to retransmit a packet multiple times without receiving any acknowledgment.

#### 5.2.2.3.6 Determinism

The adaptation layer is responsible for providing deterministic behavior for applications that demand it. This includes providing mechanisms to ensure that data is delivered with guaranteed upper bounds on latency and possibly jitter, all while maintaining coexistence between deterministic flows and best-effort traffic.

#### 5.2.2.3.7 Scheduling Mechanisms

It is envisioned that multiple different scheduling mechanisms may be employed and even coexist in the same network. This includes centralized mechanisms, for example, where a Path Computation Element (PCE) takes control of the schedule, in addition to distributed mechanisms where, for instance, neighboring nodes monitor the amount of traffic and adapt the number of cells autonomously by negotiation of the allocation or deallocation of cells as needed. The adaptation layer needs to provide mechanisms to allow for all these functions.

#### 5.2.2.3.8 Secure Communication

TSCH defines mechanisms for encrypting and authenticating frames, but it does not define how the security keys are to be generated. Hence, the adaptation layer is responsible for generating the keys and defining the authentication mechanisms by which a new node can join an existing TSCH network. The layer is also expected to provide mechanisms for the secure transfer of signaling (i.e., control) as well as application data between nodes.

The envisioned 6TiSCH protocol stack is depicted in the figure below (Fig. 5.14). RPL will be the routing protocol of choice for the architecture. As the work in IETF progresses, there may be a need to define a new 6TiSCH-specific objective function for RPL. For the management of devices, the architecture will leverage the Constraint

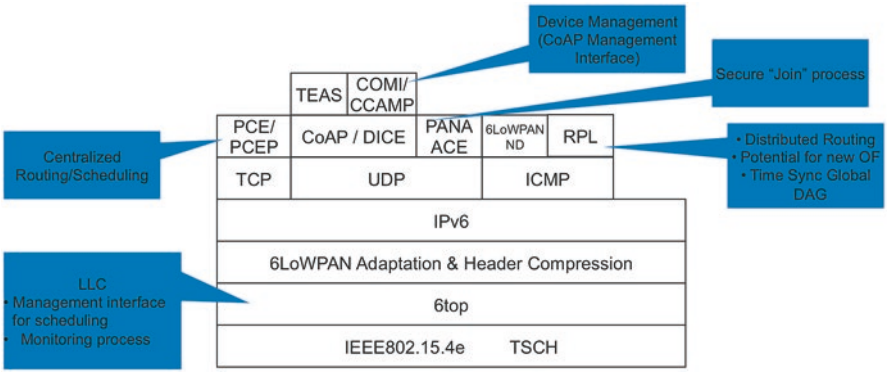



**Fig. 5.14** 6TiSCH protocol stack

Application Protocol Management Interface (COMI), which will provide the data model for the 6top adaptation layer management interface. Centralized scheduling will be carried out by the Path Computation Element (PCE). The topology and device capabilities will be exposed to the PCE using an extension to a Traffic Engineering Architecture and Signaling (TEAS) protocol. The schedule computed by the PCE will be distributed to the devices in the network using either a light-weight Path Computation Element Protocol (PCEP) or an adaptation of Common Control and Measurement Plane (CCAMP) formats. The Datagram Transport Layer Security in Constrained Environments (DICE) can be used in the architecture to secure CoAP messages. Also, the Protocol for Carrying Authentication for Network Access (PANA) will secure the process of a node joining an existing network.

## 5.3 Application Protocols Layer

Application protocols are responsible for handling the communication between Application Entities, i.e., things, gateways, and applications. They typically support the flow of data (e.g., readings or measurements) from things to applications and the flow of command or control information (e.g., to trigger or actuate end devices) in the reverse direction. These protocols define the semantics and mechanisms for message exchanges between the communicating endpoints.

The landscape of the application protocols layer in IoT is currently crowded with competing protocols and standards, each having its own set of strengths and weaknesses and with no clear path toward convergence being agreed upon by the industry yet. In this section, we will discuss the characteristics and attributes of the protocols in this layer as they pertain to IoT and will highlight, where applicable, the requirements and challenges that IoT applications impose on these protocols.

### 5.3.1 Data Serialization Formats

Applications protocols vary in the data serialization formats used to encode information into messages. One of the challenges in IoT data serialization formats is mapping between the formats used in constrained devices and those used by applications in the World Wide Web. These applications should be able to interpret the data from IoT devices with minimal format translations and a priori knowledge. Hence, the formats should be general and compatible with Web technologies. Popular data serialization formats on the Web include XML, JSON, and EXI.

Another challenge in IoT data serialization formats is the impact they have on device resource utilization, especially in terms of energy consumption. Data formats have an effect on device resource usage in two facets: in their local processing demands and their communication efficiency. The local processing demands include both the processing required to serialize memory objects into data encoded in messages and the processing required to parse the encoded messages into memory objects. The communication efficiency is a function of the compactness of the data serialization format and its efficiency to encode information in the least amount of message real estate. Both of these facets, namely, local processing and communication, have a direct impact on the energy consumption of the IoT device. Research in wireless sensor networks suggests "communication is over 1,000 times more expensive in terms of energy than performing a trivial aggregation operation". Therefore, the data serialization formats for IoT application protocols should be chosen such that they require minimal processing and communication demands.

A third challenge in IoT data serialization formats is the impact they have on network bandwidth utilization. This ties back to the compactness of the format and its encoding efficiency, as discussed above. The more verbose that the data format is, the more message space that it will consume on the wire to carry the same amount of information, which leads to less efficient use of network bandwidth. For IoT, especially when devices are connected over low-bandwidth wireless links, the data serialization format of application protocols should be chosen carefully to maximize the use of the available bandwidth.

### 5.3.2 Communication Paradigms

Application protocols support different communication patterns. These patterns enable varying paradigms of interaction between IoT applications and devices.

#### 5.3.2.1 Request/Response Versus Publish/Subscribe

The request/response paradigm enables bidirectional communication between endpoints (Fig. 5.15). The initiator of the communication sends a request message, which is received and operated upon by the target endpoint. The latter then sends a

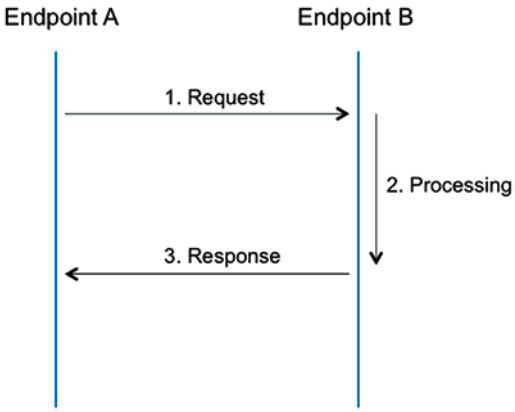



**Fig. 5.15** Request/response paradigm

response message to the original initiator. This paradigm is well suited for IoT deployments that have one or more of the following characteristics:

- The deployment follows a client-server architecture.
- The deployment requires interactive communication: both endpoints have information to send to the other side.
- The receipt of information needs to be fully acknowledged (e.g., for reliability).

However, not all IoT deployments have the above characteristics. In particular, in many scenarios, all what is required is one-way communication from a data producer (e.g., a sensor) to a consuming entity (the application). For this, the request/response paradigm is sub-optimal due to the overhead of the unneeded messages running in the reverse direction. This is where the publish/subscribe pattern comes in (Fig. 5.16).

The publish/subscribe paradigm, often referred to as pub/sub, enables unidirectional communication from a publisher to one or more subscribers. The subscribers declare their interest in a particular class or category of data to the publisher. When the publisher has new data available from that class, it pushes it in messages to interested subscribers. Besides the obvious proclamation that this paradigm optimal for IoT applications requires one-way communication, the pub/sub model is well suited for IoT deployments that can benefit from the following characteristics:

- Loose coupling between the communicating endpoints, especially when compared with the client-server model.
- Better scalability by leveraging parallelism and the multicast capabilities of the underlying transport network.

#### 5.3.2.2 Blocking Versus Non-blocking

Application protocols can offer IoT endpoints blocking or non-blocking messaging service.

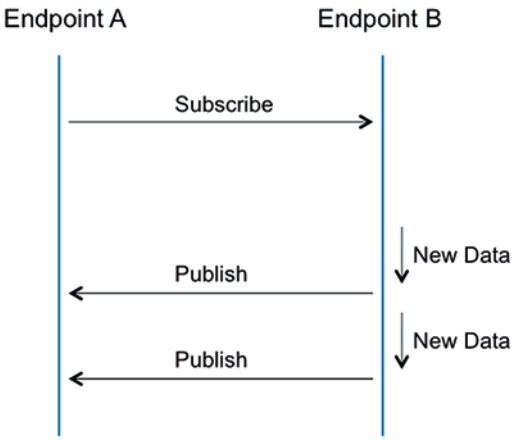



**Fig. 5.16** Publish/subscribe paradigm

In the blocking mode, the endpoint originating a request must wait to get a response to its request, after the requested operation has finished on the other endpoint. This involves potentially long or unknown wait times (where a pending request has not been responded to) for the originator.

In the non-blocking mode, the endpoint originating a request does not wait until the other endpoint has fully serviced the request. Rather, it expects a prompt acknowledgment of the request together with a specified reference, so that the originator can retrieve the outcome of the requested operation at a later point of time.

In the synchronous case, the originator of a request is not able to receive asynchronous messages, i.e., all exchanges of information between the originator and the receiver need to be initiated by the originator. The later retrieval of the result of a requested operation is through the exchange of request/response messages between the originator and the receiver.

In the asynchronous case, the originator of a request is able to receive notification messages, i.e., the receiver can send an unsolicited message to the originator at an arbitrary time to report the requested operation. The mechanisms for the notification to the originator are the same as in the case of a notification after a subscription.

### *5.3.3 QoS*

Application protocols should provide mechanisms for fine-grained control over the real-time behavior, dependability, and performance of IoT applications by means of a rich set of QoS policies. These policies should provide control over local resources and the end-to-end properties and characteristics of data transfer. The local properties controlled by QoS relate to resource usage, whereas the end-to-end properties relate to the temporal and spatial aspects of data communication.

#### 5.3.3.1 Resource Utilization

Application protocols should provide QoS policies to control the amount of memory and processing resources that can be used by the application protocol for data transmission and reception. These policies include:

##### 5.3.3.1.1 Resource Limits Policy

This policy allows control of the amount of message buffering performed by a protocol implementation, as this impacts the amount of memory consumed by that protocol. Such controls are particularly important for embedded applications running on constrained devices.

##### 5.3.3.1.2 Time Filter Policy

This policy allows applications to specify the minimum inter-arrival time between data samples. Samples that are produced at a faster pace are not delivered. This policy allows control of both network bandwidth and memory and processing power for applications which are connected over limited bandwidth networks and which might have limited computing resources.

#### 5.3.3.2 Data Timeliness

Application protocols should provide a set of QoS policies that allow control of the timeliness properties of distributed data. Specifically, the QoS policies that are desirable are described below:

##### 5.3.3.2.1 Deadline Policy

This QoS policy allows an application to define the maximum inter-arrival time for data. Missed deadline can be notified by the protocol to the application.

##### 5.3.3.2.2 Latency Budget Policy

This QoS policy provides a means for the application to communicate to the application protocol the level of urgency associated with a data communication. The latency budget specifies the maximum amount of time that should elapse from the instance when the data is transmitted to the instance when the data is placed in the queue of the associated recipients.

#### 5.3.3.3 Data Availability

Application protocols should provide the following QoS policies to allow control of data availability:

##### 5.3.3.3.1 Durability Policy

This QoS policy provides control over the degree of persistence of the data being transmitted by the application. At one end of the spectrum, it allows the data be configured to be volatile, while at the other end, it allows for data persistency. It is worth noting that data persistence enables time decoupling between the producing and the consuming endpoint by making the data available for late-joining consumers or even after the producer has disconnected.

##### 5.3.3.3.2 Life Span Policy

This QoS policy allows control of the interval of time for which a data sample will be valid.

##### 5.3.3.3.3 History Policy

This QoS policy provides a means to control the number of data samples that have to be kept available for the recipients. Possible values are the last sample only, the last N samples, or all the samples.

#### 5.3.3.4 Data Delivery

Application protocols should provide QoS policies to allow control of how data is delivered.

##### 5.3.3.4.1 Reliability Policy

This QoS policy allows the application to control the level of reliability associated with data diffusion. The possible choices are reliable and best-effort distribution. With reliable distribution, the application protocol must ensure message delivery and handle acknowledgments and retransmissions without direct application involvement.

5.3.3.4.2 Transport Priority

This QoS policy allows the application to take advantage of transports that are capable of sending messages with different priorities. Application protocols are responsible for interacting with the underlying transport layer in order to map this QoS policy to the right underlying transport network QoS markings (e.g., IP DSCP, TOS, or PCP).

### *5.3.4 RESTful Constraints*

Some application protocols adhere to a set of constraints defined by the representational state transfer (REST) architectural paradigm. REST is a distributed client-server software architecture style that was coined by Roy Fielding after he analyzed the design principles that contributed to the success of the Hypertext Transfer Protocol (HTTP) employed in the World Wide Web. Fielding concluded on a set of constraints that collectively define the REST architectural style and yield a system that is simple, scalable, and reliable.

The formal REST constraints are:

**Client-Server Communication Model**
This allows for separation of concerns where the server focuses on functions such as data storage, whereas clients focus on the user interface and user state. Uniform interfaces separate the clients from the servers. This allows for independent development of servers and clients as long as they honor the same interface.

**Stateless Communication**
The server must not store any client context that persists between requests. Session state is maintained by the client, which passes all the information necessary to service a particular request in the request itself. In other words, requests are self-contained from a server perspective.

**Cacheable Communication**
Responses from the server may be cacheable by clients and intermediate nodes. This improves the scalability and performance of the system by partially or completely eliminating some client-server interactions.

**Layered Architecture**
To allow for better scalability, the system comprises of a layered architecture that includes clients, servers, and potentially multiple intermediate nodes interspersed between them. Clients may be in communication with intermediate nodes or directly with servers without ordinarily being able to identify a difference between the two.

**Uniform Interfaces**
All interactions between clients and servers (or intermediate nodes) are governed by uniform interfaces. These interfaces use the notion of "resources." A resource is an abstraction for server-side information and associated native data representation. Resources have unique identifiers (e.g., URIs in Web systems). When a server

communicates with a client, it transfers an external representation of the resource to the client (hence the name representational state transfer). REST interfaces are representation centric. Hence, a small set of operations (also called verbs), which are uniform across all use cases, can be used in the interface. Usually, this set of verbs is referred to as CRUD for create, read, update, and delete. In REST interfaces, there is no out-of-band contract that defines the types of actions that can be initiated by a client. Rather, this information is discovered dynamically by the client from prior server interactions through hypermedia (i.e., by hyperlinks within hypertext). This characteristic of the interface is known as hypermedia as the engine of application state (HATEOAS).

**Code on Demand**
Client functionality may be extended or modified by the server through the transfer of executable pieces of code that can be executed on the client side (e.g., scripts or applets). This is an optional REST constraint known as "code on demand."

## *5.3.5 Survey of IoT Application Protocols*

### 5.3.5.1 CoAP

The Constrained Application Protocol (CoAP) was standardized by the IETF Constrained RESTful Environments (CORE) workgroup as a lightweight alternative to HTTP, targeted for constrained nodes in low-power and lossy networks (LLNs). The need for a lighter-weight version of HTTP can be appreciated by examining, for example, the number of messages that need to be exchanged between a client and a server to perform a simple Get operation on a resource: first there are three TCP SYN messages exchanged to bring up the TCP session, followed by the HTTP Get request from the client, then the HTTP response from the server, and finally two messages to terminate the TCP session. Hence, a total of seven messages are required just to fetch a resource. CoAP reduces this overhead by using UDP as a transport in lieu of TCP. CoAP also uses short headers to reduce message sizes.

Similar to HTTP, CoAP is a RESTful protocol. It supports the create, read, update, and delete (CRUD) verbs but in addition provides built-in support for the publish/subscribe paradigm via the new observe verb. CoAP optionally provides a mechanism where messages may be acknowledged for reliability and provides a bulk transfer mode. CoAP was standardized as RFC 7252. Furthermore, there is an ongoing work in the IETF to define mechanisms for dynamic resource discovery in CoAP via a directory service.

### 5.3.5.2 XMPP

The Extensible Messaging and Presence Protocol (XMPP) was originally designed for instant messaging, contact list, and presence information maintenance. It is a message-centric protocol based on the Extensible Markup Language (XML). Due

to its extensibility, the protocol has been used in several applications, including network management, video, voice-over IP, file sharing, social networks, and online gaming, among others. In the context of IoT, XMPP has been positioned for smart grid solutions, for example, as depicted in RFC 6272. XMPP originally started as an open-source effort, but the core protocol was later standardized by the IETF in RFC 6120 and 6121. Moreover, the XMPP Standards Foundation (XSF) actively develops open extensions to the protocol.

The native transport protocol for XMPP is TCP. However, there is an option to run XMPP over HTTP.

#### 5.3.5.3 MQTT

The Message Queue Telemetry Transport (MQTT) protocol is a lightweight publish/subscribe messaging protocol that was originally designed by IBM for enterprise telemetry. MQTT follows a client-server architecture where clients connect to a central server (called the broker). The protocol is message oriented, where messages are published to an address, referred to as a topic. Clients subscribe to one or more topics and receive updates from a client that is publishing messages for this topic. In MQTT, topics are hierarchical (similar to URLs), and subscriptions may use wildcards. MQTT is a binary protocol, and it uses TCP transport. The protocol is being standardized by the Organization for the Advancement of Structured Information Standards (OASIS).

The protocol targets endpoints where "a small code footprint" is required or where network bandwidth is limited; hence it could prove useful for constrained devices in IoT.

#### 5.3.5.4 AMQP

The Advanced Message Queuing Protocol (AMQP) originates from financial sector applications but is generic enough to accommodate other types of applications. AMQP is a binary message-oriented protocol. Due to its roots, AMQP provides message delivery guarantees for reliability, including at least once, at most once, and exactly once. The importance of such guarantees can be easily seen in the context of financial transactions (e.g., when executing a credit or debit transaction). AMQP offers flow control through a token-based mechanism, to ensure that a receiving endpoint is not overburdened with more messages than it is capable of handling. AMQP assumes a reliable underlying transport protocol, such as TCP.

AMQP was standardized by OASIS in 2012 and then by the International Standards Organization (ISO) and the International Electrotechnical Commission (IEC) in 2014. Several open-source implementations of the protocol are available. AMQP defines a type system for encoding message data as well as annotating this data with additional context or metadata. AMQP can operate in simple peer-to-peer mode as well as in hierarchical architectures with intermediary nodes, e.g., messag-

ing brokers or bridges. Finally, AMQP supports both point-to-point communication and multipoint publish/subscribe interactions.

#### 5.3.5.5 SIP

The Session Initiation Protocol (SIP) handles session establishment for voice, video, and instant messaging applications on IP networks. It also manages presence (similar to XMPP).

SIP invitation messages used to create sessions carry session descriptions that enable endpoints to agree on a set of compatible media types. SIP leverages elements called proxy servers to route requests to the user's current location, authenticate and authorize users for services, implement call-routing policies, and provide features. SIP also defines a registration function that enables users to update their current locations for use by proxy servers. SIP is a text-based protocol and can use a variety of underlying transports, TCP, UDP, or SCTP, for example. SIP is standardized by the IETF as RFC 3261.

#### 5.3.5.6 IEEE 1888

IEEE 1888 is an application protocol for environmental monitoring, smart energy, and facility management applications. It is a simple protocol that supports reading and writing of time-series data using the Extensible Markup Language (XML) and the simple object access protocol (SOAP). The data is identified using Universal Resource Identifiers (URIs). The latest revision of the protocol was standardized by the IEEE Standards Association in 2014.

#### 5.3.5.7 DDS RTPS

Distributed Data Service Real Time Publish and Subscribe is a data-centric application protocol that, as its name indicates, supports the publish/subscribe paradigm. DDS organizes data into "topics" that listeners can subscribe to and receive asynchronous updates when the associated data changes. DDS RTPS provides mechanisms where listeners can automatically discover speakers associated with specific topics. IP multicast or a centralized broker/server may be used to that effect. Multiple speakers may be associated with a single topic and priorities can be defined for different speakers. This provides a redundancy mechanism for the architecture in case a speaker fails or loses communication with its listeners.

DDS RTPS supports very elaborate QoS policies for data distribution. These policies cover reliability, data persistence, delivery deadlines, and data freshness. DDS RTPS is a binary protocol, and it uses UDP as the underlying transport. The latest version of the protocol was standardized by the Object Management Group

**Table 5.1** Survey of IoT application protocols

| Protocol | Functions | Primary use | Transport | Format | SDO |
|---|---|---|---|---|---|
| CoAP | REST resource manipulation via CRUD<br>Resource tagging with attributes<br>Resource discovery through RD | LLNs | UDP | Binary | IETF |
| XMPP | Manage presence<br>Session establishment<br>Data transfer (text or binary) | Instant messaging | TCP<br>HTTP | XML | IETF<br>XSF |
| MQTT | Lightweight pub/sub messaging<br>Message queuing for future subscribers | Enterprise telemetry | TCP | Binary | OASIS |
| AMQP | Message orientation, queuing and pub/sub<br>Data transfer with delivery guarantees (at least once, at most once, exactly once) | Financial services | TCP | Binary | OASIS |
| SIP | Manage presence<br>Session establishment<br>Data transfer (voice, video, text) | IP telephony | TCP, UDP, SCTP | XML | IETF |
| IEEE 1888 | Read/write data into URI<br>Handling time-series data | Energy and facility management | SOAP/ HTTP | XML | IEEE |
| DDS (RTPS) | Pub/sub messaging with well-defined data types<br>Data discovery<br>Elaborate QoS | Real-time distributed systems (military, industrial, etc.) | UDP | Binary | OMG |

(OMG) in 2014. The table below provides a summary of the protocols discussed in this section (Table 5.1).

## 5.4 Application Services Layer

### *5.4.1 Motivation*

M2M deployments have existed for over two decades now. However, what has characterized these deployments is a state of fragmentation: vertical solutions are implemented in silos with proprietary communication stacks and very tight coupling between applications and devices. The paradigm can be best described as "one application-one device." The application code is exposed to all the device specifics under this modus operandi. This, in turn, creates complexity and increases the cost of the solution's initial development and ongoing maintenance. For instance, if the

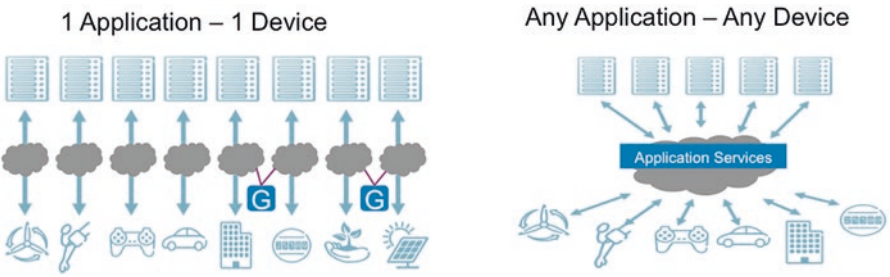



**Fig. 5.17** Application to device coupling

layer of abstraction that fits in between the applications and the devices, i.e., things, and enables the paradigm of "any application-any device" (Fig. 5.17).

In other words, this abstraction layer provides a common set of services that enables an application to interface with potentially any device without understanding a priori the specifics and internals of that device. This abstraction layer is referred to as the Application Services layer in our model of the IoT protocol stack. It provides seamless interoperability between applications and devices and promotes nimble development of IoT solutions.

From a business perspective, the emergence of this new layer is driven, in part, by communication service providers (CSPs) looking at using IoT to gain additional revenue from their networks. Key to this revenue will be differentiating beyond providing simple IP connectivity. CSPs know well the value of IoT is in the data, not the way it is transported. To unlock this value, the Application Services layer aims to turn the network to a common platform to enable diverse IoT applications. This common platform will be built across an ecosystem of heterogeneous devices and will enable CSPs to monetize IoT data access, storage, management, and security.

### *5.4.2 Industry Progress*

In 2012, the European Telecommunications Standards Institute (ETSI) published the first release of its M2M service layer standard defining a standardized platform for multiservice IoT solution. Later that year, seven standards development organizations (TIA and ATSI from the United States, ARIB and TTC from Japan, CCSA from China, ETSI from Europe, and TTA from Korea) launched a global organization to jointly define and standardize the common horizontal functions of the IoT Application Services layer under the umbrella of the oneM2M Partnership Project (http://www.onem2m.org). The founders agreed to transfer and stop their own overlapping IoT application service layer work.

In what follows, we will discuss the ETSI M2M and oneM2M efforts in more details.

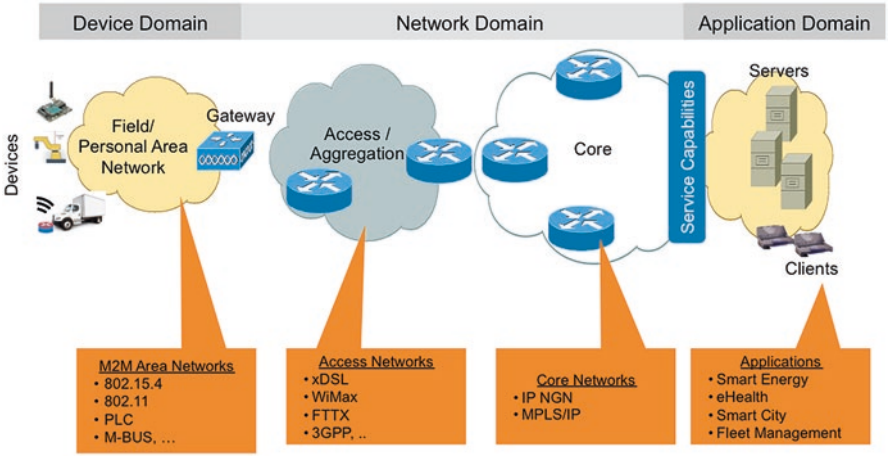



**Fig. 5.18** ETSI M2M network architecture

### 5.4.2.1 ETSI M2M

The network architecture adopted by the ETSI M2M effort draws heavily on existing technologies. The architecture comprises of three domains: M2M device domain, network domain, and application domain (Fig. 5.18). The M2M device domain provides connectivity between things and gateways, e.g., a field area network or personal area network. Devices are entities that are capable of replying to request for data contained within those entities or capable of transmitting data contained within those entities autonomously. Gateways ensure that end devices (which may not be IP enabled) can interwork and interconnect with the communication network. Technologies in the M2M device domain include IEEE 802.15.4, IEEE 802.11, Zigbee, Z-WAVE, PLC, etc.

The network domain includes the communication networks, which interconnect the gateways and applications. This typically includes access networks (xDSL, FTTX, WiMax, 3GPP, etc.) as well as core networks (MPLS/IP). The application domain includes the vertical-specific applications (e.g., smart energy, eHealth, smart city, fleet management, etc.) in addition to the Service Capabilities layer (SCL), a middleware layer that provides various data and application services. The main focus of the ETSI M2M standards is on defining the functionality of the SCL. The SCL provides functions that are common across different applications and exposes those functions through an open API. The goal is to simplify application development and deployment through hiding the network specifics.

The functions of the SCL may reside on entities deployed in the field such as devices and gateways or on entities deeper in the network (e.g., servers in a data center). This gives rise to three flavors of SCL, depending on its placement: device SCL (D-SCL), gateway SCL (G-SCL), and network SCL (N-SCL). While the three flavors of SCL do share some common functions, they also differ due to the differ-

ent operations that need to be carried out by devices, gateways, and network nodes (servers). In general, the SCL provides the following functions:

- Registration of devices, applications, and remote SCLs
- Synchronous and asynchronous data transfer
- Identification of applications and devices
- Group management for bulk endpoint addressability and operations
- Security mechanisms for authentication, authorization, and access rights control
- Remote device management (through existing protocols)
- Location information

ETSI M2M adopted a RESTful architecture style where all data in the SCL is represented as resources. This includes not only the data generated by the devices but also data representing device information, application information, remote SCL information, access rights information, etc. Resources in the SCL are uniquely addressable via Universal Resource Identifiers (URIs). Manipulation of the resources is done through a RESTful API, which provides the CRUD primitives (C, create; R, read, U, update, D, delete). The API can be bound to any RESTful protocol, such as HTTP or CoAP. ETSI technical specification TS 102 921 specifies the API binding to HTTP and CoAP protocols.

Resources within the SCL are organized in a well-specified hierarchical structure known as the resource tree (Fig. 5.19). This provides a number of advantages: it provides a data mediation function, describes how resources relate to each other, allows traversal and query of data in an efficient manner, and speeds up the development of platforms. The resource tree of an SCL includes:

- Location of other SCLs in the network (in other devices or GWs)
- List of registered applications

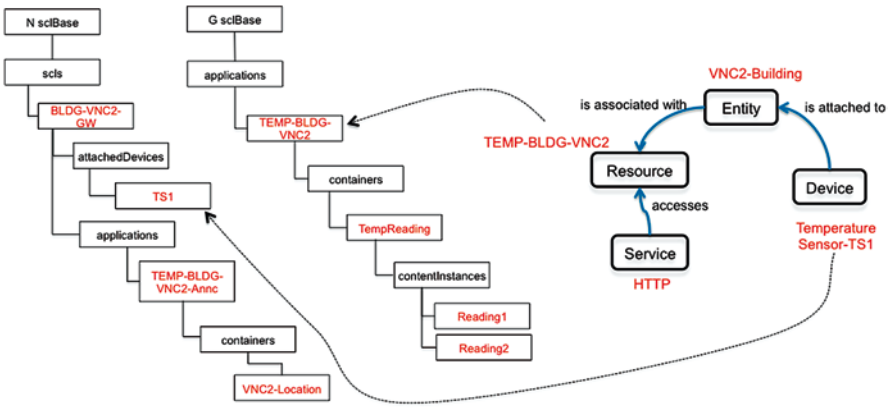



**Fig. 5.19** Example ETSI M2M resource tree

- Announced resources on remote elements
- Access rights to various resources
- Containers to store actual application data

In addition to the different flavors of SCL, ETSI M2M defines the following types of entities: application and devices. Applications are further categorized as network applications (NA), gateway applications (GA), or device applications (DA) depending on whether they run in the network domain, on a gateway or embedded on a device, respectively. Devices are categorized into those that support the ETSI SCL functions (known as D devices) and those that do not support these functions (known as D devices).

ETSI M2M defines a number of reference points, or interfaces, between interacting entities. These reference points define the semantics of the interactions, and associated API, between the entities. In particular, the following three reference points are defined:

- mIa: defines the interactions between a network application and the N-SCL. Allows the application to register with the SCL and access resources on it,
- mId: defines the interactions between a device application, on one hand, and a D-SCL or G-SCL on the other. Allows the application to register with the SCL and access resources on it,
- dIa: defines the interactions between the N-SCL, on one hand, and the D-SCL or G-SCL on the other. Allows the various SCL instances to register with one another and access their respective resources.

The ETSI M2M architecture supports backward compatibility with devices that do not support the ETSI reference point functions. This compatibility is achieved

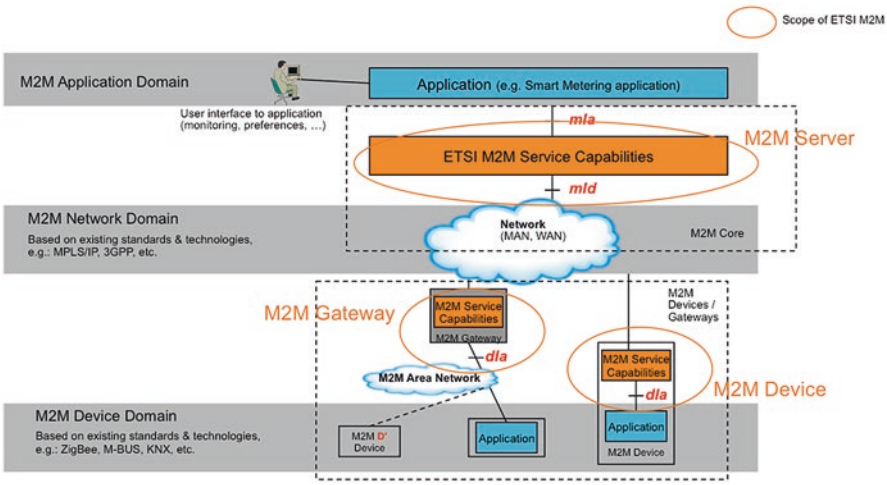



**Fig. 5.20** ETSI M2M system architecture

through gateways that communicate with the legacy devices via their own proprietary mechanisms and handle the translation of the data into the resource tree. ETSI does not define the specifics of how the translation should be performed (Fig. 5.20).

Irrespective of the underlying physical network topology, the ETSI model defines a strict two-level hierarchy with N-SCL at the top level and G-SCL or D-SCL at the bottom level. The daisy chaining of SCLs in deeper hierarchies is not defined or supported.

The ETSI M2M functional architecture is defined in technical specification TS 102 690.

#### 5.4.2.2 oneM2M

The oneM2M standards consider any IoT deployment to be comprised of two domains: the field domain and the infrastructure domain (Fig. 5.21). The field domain includes things (e.g., sensors, actuator, etc.) and gateways, whereas the infrastructure domain includes the communication networks (aggregation, core) as well as the data centers. From a functional perspective, each of these domains includes three flavors of entities: an application entity, a common services entity, and a network services entity.

The application entity implements the vertical-specific application logic. It may reside on one or multiple physical nodes in the deployment. Examples of an application entity would be a home automation application or a smart parking application.

The common services entity is a middleware layer that sits in between applications (application entity) and the underlying network services (network services entity) (Fig. 5.22). The common services entity (CSE) provides the following set of common functions to applications:

- Identity management: Identification of applications entities and CSEs.
- Registration: Includes registration of application entities and CSEs.
- Connectivity handling: This ensures efficient, reliable, and scalable use of the underlying network.

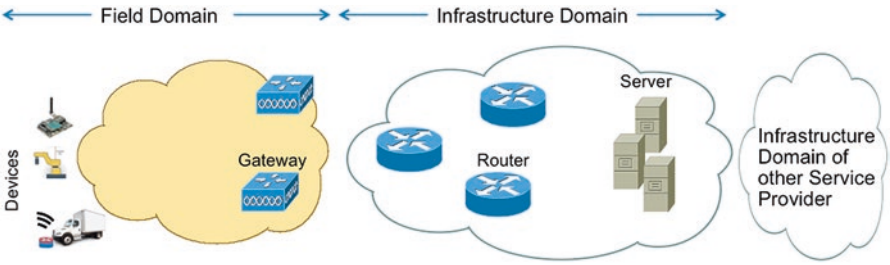



**Fig. 5.21** oneM2M domains

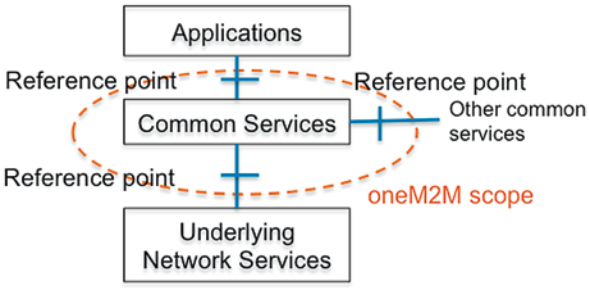



**Fig. 5.22** oneM2M common services entity

- Remote device management: This includes configuration and diagnostic functions.
- Data exchange: Supports storing and sharing of data between applications and devices, in addition to event notification.
- Security and access control: Provides control over access to data (who can access what and when, etc.).
- Discovery: Provides discovery of entities as well as data and resources.
- Group management: Support of bulk operations and access.
- Location: Provides an abstraction for managing and offering location information services.

The CSE is, more or less, logically equivalent to the ETSI M2M SCL.

The network services entity provides value-added services to the CSE, such as QoS, device management, location services, and device triggering.

The oneM2M reference architecture identifies five different types of logical nodes: application-dedicated nodes, application service nodes, middle nodes, infrastructure nodes, and none-oneM2M nodes. These nodes may map to one or more physical devices in the network or may have no corresponding physical mapping.

Application-dedicated nodes (ADNs) are oneM2M compliant devices (i.e., things) with restricted functionality: they include one or more application entities but no CSE. From a physical mapping perspective, ADNs may map to constrained IoT devices.

Application service nodes (ASNs) are fully featured oneM2M compliant devices. They include a CSE in addition to one or more application entities. From physical mapping standpoint, they map to (typically non-constrained) IoT devices.

Middle nodes (MNs) host a CSE. A middle node may or may not include application entities. There could be zero, one, or many middle nodes in the network. MNs physically map to gateways in the network.

Infrastructure nodes (INs) host the CSE and may or may not host any application entities. The CSE on the IN includes functions that do not typically exist in any other CSE in the network. There's a single infrastructure node per domain per service provider in the oneM2M architecture.

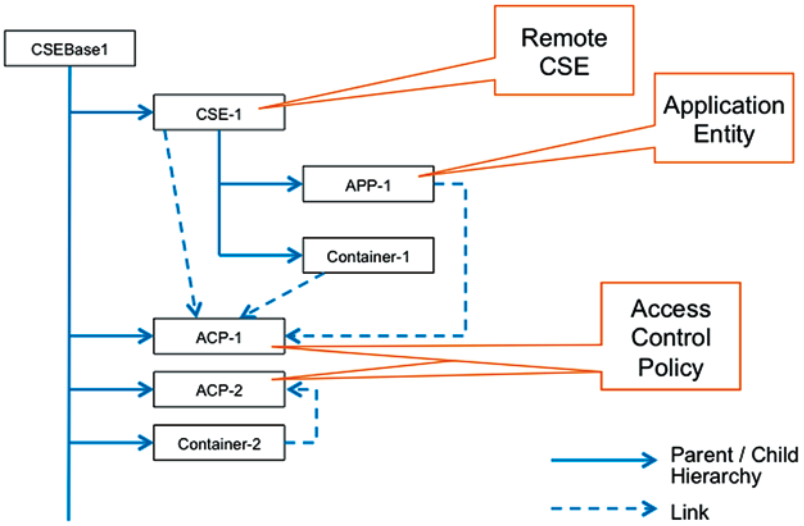



**Fig. 5.23** oneM2M resource organization

None-oneM2M Nodes are legacy devices that interwork with the oneM2M architecture. This provides backward compatibility of oneM2M with existing systems (similar to D devices in the ETSI M2M architecture).

As with ETSI M2M, oneM2M follows a RESTful architecture style where all data is modeled as resources, albeit oneM2M does not define a static resource structure like the ETSI resource tree. Instead, the standard provides means by which resources can be linked together (through resource links). Client applications can discover the resource organization dynamically. In this regard, the oneM2M approach complies with the HATEOAS (Hypermedia as the Engine of Application State) REST constraint discussed in Sect. 5.3.4, because it does not assume that the clients have any a priori knowledge of the resource organization (Fig. 5.23).

Similar to ETSI M2M, oneM2M defines a set of reference points or interfaces between interacting entities. The oneM2M standard defines the following four reference points:

- Mca: Defines the interactions between application entities and CSE.
- Mcn: Defines the interactions between the CSE and the underlying network service entity.
- Mcc: Defines the interactions between two CSEs in the same service provider domain.
- Mcc': Defines the interactions between two CSEs across service provider domain boundary.

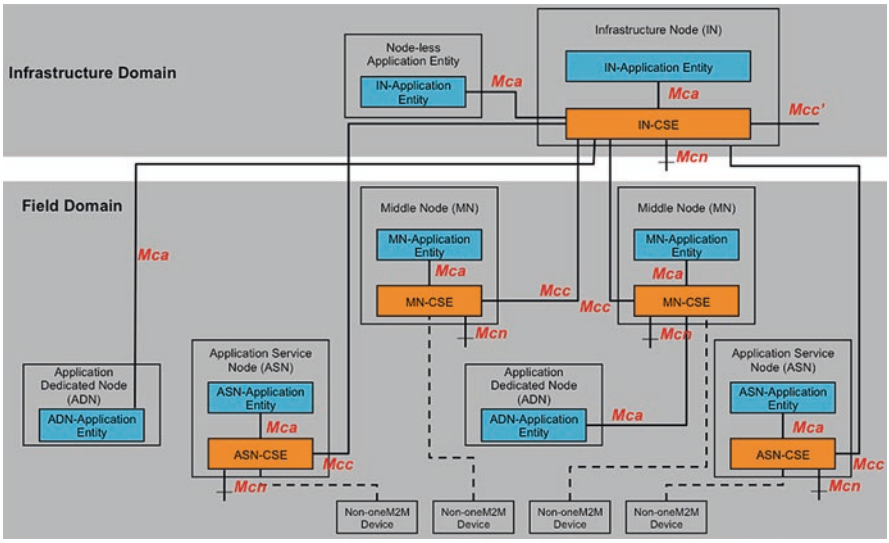



**Fig. 5.24** oneM2M functional architecture

A number of notable differences between the reference points defined by ETSI M2M and those defined by oneM2M are worth highlighting:

First, ETSI M2M defines two different reference points for interactions between applications and the middleware as well as between devices and the middleware (mIa and mId interfaces, respectively), whereas oneM2M collapses both interfaces into the Mca reference point.

Second, the Mcn reference point is unique to oneM2M and has no equivalent in the ETSI standard. This interface enables the middleware to access network service functions. For example, it can be used to signal information from the service layer to the transport layer to request QoS and prioritization for M2M communication, for transmission scheduling, to signal indication for small data transmission, for device triggering, etc.

The interface may also be used to extract information from the underlying transport layer, for example, to fetch data related to the location of M2M devices or gateways (Fig. 5.24).

### *5.4.3 Technology Gaps*

While ETSI and oneM2M have made strides in defining standard APIs and common application services for IoT, several gaps remain.

First, in terms of search and discovery capabilities, the IoT Application Services layer should provide support for:

- Mechanisms by which devices as well as applications can automatically discover each other as well as discover middleware/common services nodes.
- Mechanisms by which applications can search for devices with specific attributes (e.g., sensors of particular type) or context (e.g., within a specific distance from a location).
- Mechanisms by which applications can search for data based on attributes (e.g., semantic annotations) or context (e.g., spatial or temporal).

Both ETSI and oneM2M define basic mechanisms for resource search based on metadata or text strings. However, these are rudimentary capabilities and do not provide the contextual search functions that will be needed for IoT. Furthermore, no mechanisms for device or gateway auto-discovery are provided by either standard. It is assumed that the various instances of the middleware (SCL in case of ETSI and CSE in case of oneM2M), which need to communicate with each other, have a priori knowledge of their respective IP addresses. The same assumption holds between application endpoints and other entities (devices or middleware instances) that they need to communicate with.

Second, with regard to data encoding, interpretation, and modeling, the Application Services layer should encompass:

- Mechanisms that render IoT data understandable to applications without a priori knowledge of the data or the devices that produced it.
- Mechanisms that enable application interaction at a high level of abstraction by means of physical/virtual entity modeling.
- Mechanisms that enable data management services to host the semantic description of IoT data that is being handled.
- Framework for defining formal domain-specific semantic models or ontologies, including but not limited to defining an upper-level ontology for IoT.

ETSI's effort stopped at defining opaque containers for holding data. The interpretation of that data was outside the scope of what was standardized. OneM2M went one step further by providing an attribute to link the data container to an ontology reference (URI). However, no formal effort has been undertaken to define any ontologies or define any associated framework for tying semantic systems with the rest of the architecture, beyond this simple linkage.

## 5.5 Summary

In this chapter we started with an overview of the IoT protocol stack, and then we examined each of the Link layer, Internet layer, Application Protocols layer, and Application Services layer in details. For each of these layers, we examined the IoT

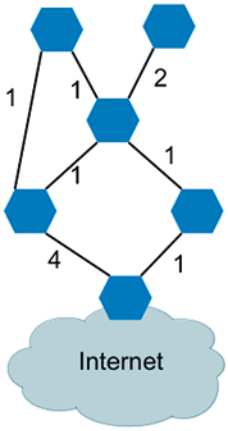
challenges and requirements impacting the protocols, which operate at that respective layer, and discussed the industry progress and gaps.

In the course of the discussion on the Link layer, we covered IEEE 802.15.4, TCSH, IEEE 802.11ah, and Time-Sensitive Networking (TSN). In the Internet layer, we discussed 6LowPAN, RPL, and 6TiSCH. In the Application Protocols layer, we surveyed a subset of the multitude of available protocols. Finally, in the Application Services layer, we covered the work in ETSI M2M and oneM2M on defining standard application middleware services.

## Problems and Exercises

1. What is the difference between IEEE 802.15.4 full-function device (FFD) and reduced-function device (RFD)?
2. IEEE 802.11ah and IEEE 802.15.4 both provide a low-power wireless protocol. What are the main differences between the two?
3. Why does IEEE 802.1Qca use IS-IS as the underlying protocol and not some other routing protocols such as OSPF or BGP?
4. What are three functions provided by the 6LowPAN adaptation layer?
5. Is RPL a link-state or distance-vector routing protocol? Why did the IETF ROLL workgroup decide to go with that specific flavor of routing protocols?
6. What are the constraints that characterize the RESTful communication paradigm?
7. What is the Application Services layer in the IoT protocol stack? What services does it provide?
8. What are the functions of the Service Capabilities layer (SCL) in the ETSI M2M architecture?
9. What are functions of the common services entity (CSE) in the oneM2M architecture? How do they compare to those of ETSI's SCL?
10. Why do the IoT application services architectures under standardization all follow the RESTful paradigm?
11. A temperature sensor that supports CoAP has an operating range of 0–1000 °F reports a reading every 5 s. The sensor has a precision of 1/100 °F. The sensor reports along with every temperature reading a time stamp using the ISO 8601 format (CCYY-MM-DDThh:mm:ss).
    (a) If the current temperature measured by the sensor is 342.5 °F, construct the payload of a CoAP message with the reading encoded in XML and then in JSON.
    (b) Assuming that the sensor consumes 3 nano-Joules per byte (character) transmitted over a wireless network, calculate the total energy required to transmit each message. Which of the two encoding schemes (XML or JSON) is more energy efficient? By what percentage?

12. Compare the bandwidth utilization for the XML vs. JSON messages of Question 11 in bits per second assuming UTF-8 text encoding is being used.
13. An IoT water level monitoring application requires updates from a sensor periodically, using the command/response paradigm. The application triggers a request every 1 s. The round-trip propagation delay between the application and the sensor is 12 milliseconds. The sensor consumes 3 milliseconds on average to process each request. The application consumes 2 milliseconds to send or receive any message. If the application blocks on every request to the sensor, how much of its time budget can be saved by redesigning the application to use the publish/subscribe communication model in lieu of the command/response approach?
14. A utility company uses IPv6-enabled smart meters running in an IEEE 802.15.4 mesh. If the mesh is operating at 1Mbps without 6LoWPAN IPv6 header compression, what is the throughput of the smart metering application in the worst-case scenario?
15. An automotive parts manufacturer is looking to upgrade the network that controls their computer numerical control (CNC) mill. At full speed, the mill can cut into solid steel at a rate of 1″ per second. The manufacturer's quality assurance (QA) guideline mandates that the dimensions of any part produced must be accurate within ±1/100." In order to meet the QA guideline, what is the maximum jitter that needs to be guaranteed by the new deterministic network that connects the mill to the controlling computer?
16. Given the following IEEE 802.15.4 mesh running the RPL protocol. The numbers indicated next to each link is the associated latency. If the objective function is to minimize the communication latency to the Internet, what will be the topology computed by RPL?



17. An automation engineer is looking to deploy a deterministic network in a sheet metal factory. The control system in charge of safety expects a message from the embedded application of a heating element controller every 50 milliseconds, otherwise it immediately shuts down the production line. The network in question has on average a delay of 1 msec per link and 2 msec per node. What is the maximum number of hops that can separate the control system from the heating element controller?
18. Why does channel hopping improve the reliability of wireless sensor networks?
19. An application protocol supporting a time filter policy support for client applications must not deliver messages at a rate higher than what the client application is willing to consume. What are common strategies to achieve this?
20. Which Application layer protocol would you choose for deploying an IoT solution for a financial institution? Why?

## References


1. J. Yick et al., Wireless sensor network survey. Comput. Netw **52**(12), 2292–2330 (2008)
2. M. Sichitiu, Wireless Mesh Networks: Opportunities and Challenges, Wireless World Congress, 1–6, 2005
3. IEEE 802.15.4–2011, September 2011
4. R. Krasteva et al., Application of wireless protocols Bluetooth and ZigBee in telemetry system development. Prob. Eng. Cybern. Robot **55**, 30–38 (2005)
5. N. Garg, M. Yadav, A review on comparative study of Bluetooth and ZigBee, Proceedings of the Second International Conference on Advances in Electronics, Electrical and Computer Engineering, EEC 2013
6. IEEE 802.15.4g-2012, April 2012
7. T. Adame et al., IEEE 802.11ah: The Wi-Fi Approach for M2M Communications, IEEE Wireless Communications, December 2014
8. IEEE draft standard 802.11ah Draft 4
9. M. Teener. IEEE 802 Time Sensitive Networking: Extending Beyond AVB
10. "Industrial Ethernet: A Control Engineer's Guide", Cisco Whitepaper
11. D. Pannell, Audio Vidor Bridging Gen 2 Assumptions, July 2011
12. IEEE 802.1Qca Draft 2.0, April 2015
13. P. Meyer et al., Extending IEEE 802.1 AVB with time-triggered scheduling: A simulation study of the coexistence of synchronous and asynchronous traffic, IEEE Vehicular Networking Conference (VNC), At Boston, Massachusetts, 2013
14. IEEE 802.1Qbv Draft 2.3, April 2015
15. IEEE Standard 802.15.4e-2012
16. T. Watteyne et al., Using IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) in the Internet of Things (IoT): Problem Statement, IETF RFC 7554, May 2015
17. X. Su et al., Enabling Semantics For The Internet of Things – Data Representation and Energy Consumption, Internet of Things Finland, January 2013
18. Z. Shelby et al., The Constrained Application Protocol (CoAP), IETF RFC 7252, June 2014
19. Z. Shelby et al., CoRE Resource Directory, draft-ietf-core-resource-directory, work in progress, March 2016

20. Baker & Meyer, Internet Protocols for the Smart Grid, IETF RFC 6272, June 2011
21. J. Rosenberg et al., SIP: Session Initiation Protocol, IETF RFC 3261, June 2002
22. T. Watteyne et al., Reliability through frequency diversity: why channel hopping makes sense, PE-WASUN '09 Proceedings of the 6th ACM symposium on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks, Pages 116–123, October 2009
23. LoRa Alliance, Technical Marketing Workgroup 1.0, "LoRaWAN What is it? A technical overview of LoRa and LoRaWAN", November 2015
24. LoRaWAN Adaptive Data Rate: https://www.thethingsnetwork.org/wiki/LoRaWAN/ADR
25. F. Adelantado et al. Understanding the Limits of LoRaWAN, IEEE Communications Magazine, January 2017