MODULE - 1

1.1 What Is IoT?

A world where everything is online and communicating to other things and people to enhance people's lives likeself-driving drones and sensors for monitoring your health, is collectively known as the ***Internet of Things*** (IoT).

Basic goal of IoT : "connect the unconnected."

This means that objects not currently joined to the Internet will be connected so that they can communicate and interact with people and other objects.

- IoT allow to sense and control the physical world by making objects smarter and connecting them through an intelligent network.
- Integration between the physical world and computers allows for improvements in efficiency, accuracy, automation, and the enablement of advanced applications.

IoT should be viewed as an umbrella of various concepts, protocols, and technologies, which are designed to create new products as well as new challenges, such as scaling vast amounts of data that need to be processed.

1.2 GENESIS OF IOT

The IoT started between the years 2008 and 2009. "Internet of Things" is invented by Kevin Ashton. Kevin quoted as saying: "In the 20th century, computers were brains without senses—they only knew what we told them." Computers depended on humans to input data and knowledge. But in the 21st century, computers are sensing things too.

The evolution of the Internet can be categorized into four phases:

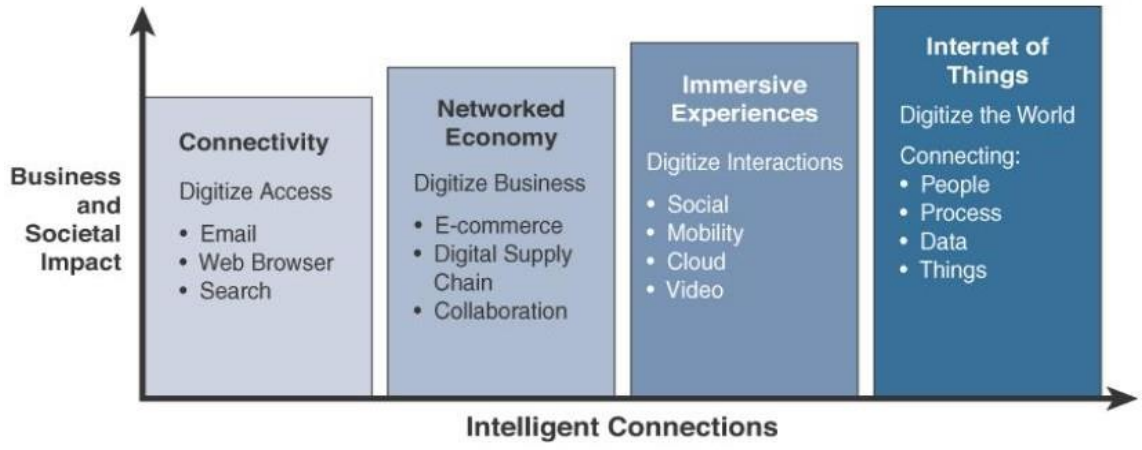


Figure 1-1 *Evolutionary Phases of the Internet*

	Internet Phases	Definition
1.	Connectivity (Digitize access)	• Connect people using email, web services… • Search and access the information
2.	Networked Economy (Digitize business)	• Enable e-commerce and supply-chain enhancements • Collaborative engagement to increase efficiency
3.	Immersive Experiences (Digitize interactions)	• Extend Internet using social media while always being connected through mobility. • Most applications are cloud-based.
4.	Internet of Things (Digitize the world)	• Connect objects and machines in real world. • Enable connecting the unconnected.

1.3 IOT AND DIGITIZATION

IoT is focused on connecting "things" to Internet.

Example: Wi-Fi location tracking in shopping mall

"things"	Wi-Fi devices
Operation	Tracking consumer location to understand how much time they spend in different parts of a mall or store through their smart phone.
Advantages	Changing locations of product displays and advertising, shops, rent to charge and security positions.

Digitization is the conversion of information into a digital format. It is focused on connecting "things" with its data and business result.

Example: Digitization of Photography

"things"	Digital camera
Advantages	No need retailer to develop film and better capturing of images.

Example: Digitization of Taxi services

"things"	Taxi Driver device, Rider mobile
Advantages	Mobile app identifies cab, driver and fare. The rider pays fare through app.

In the context of IoT, digitization brings together things, data, and business process to make networked connections more relevant and valuable.

Example: "Nest" home automation
The sensors determine desired climate settings and other smart objects, such as smoke alarms, video cameras, and other third-party devices. The devices and their functions are managed and controlled together and could provide the holistic experience.

Smart objects and increased connectivity drive digitization, and thus many companies, countries, and governments are embracing this growing trend.

1.4 IOT IMPACT

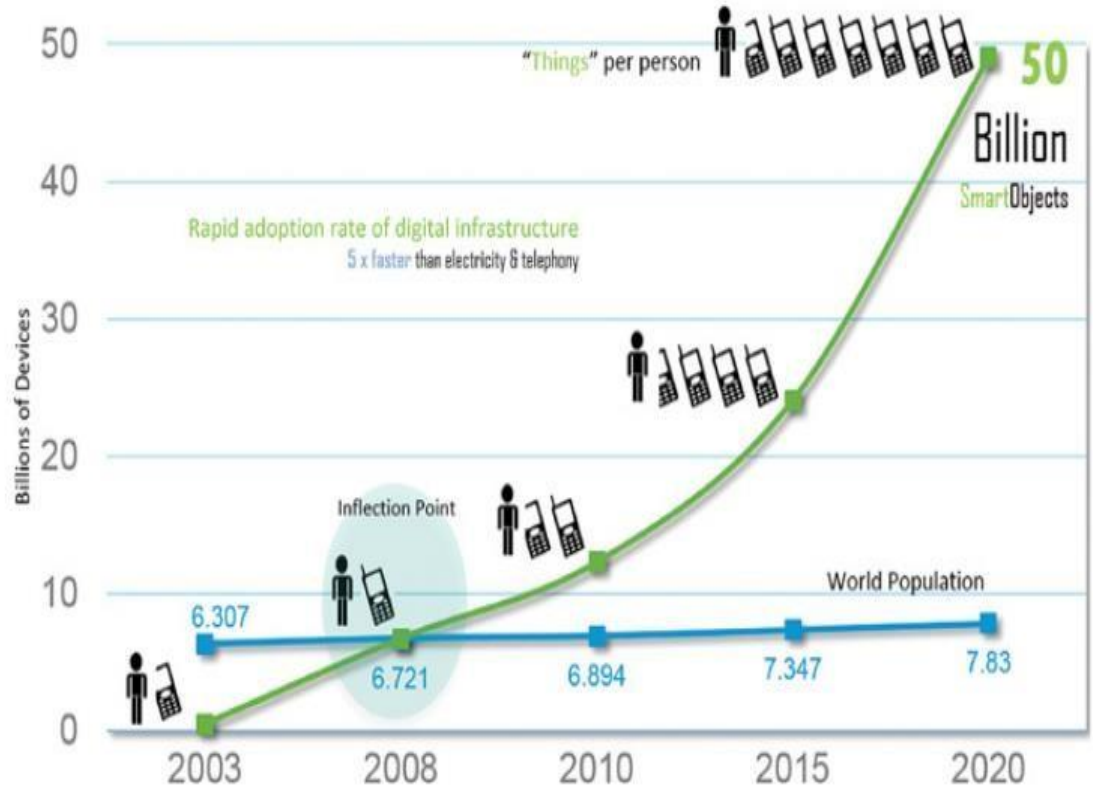


Figure 1-2 *The Rapid Growth in the Number of Devices Connected to the Internet*

While self-driving car is now becoming a reality with well-known projects like Google's self-driving car, IoT is a necessary component for implementing a fully connected transportation infrastructure.

IoT→

- provide better interaction with the transportation system through bidirectional data exchanges
- provide important data to the riders.
- provide reliable communications and data from transportation-related sensors.

A self-driving car designed by Google operations:

- Basic sensors in cars monitor oil pressure, tire pressure, temperature, and other core car functions.
- The driver can access these data while controlling the car using equipment such as a steering wheel, pedals, and so on.
- The driver will understand, handle, and make critical decisions while concentrating on driving safely.

- IP-enabled sensors allow easy communication with other systems both inside and outside the car.
- Sensors and communication technologies vehicles to "talk" to other vehicles, traffic signals, school zones, and other elements of the transportation infrastructure.

Transportation challenges can be classified into the three categories:

Challenges	**Description**
Safety	IoT enabled technologies enable drivers to avoid crashes. Reduce number of lives lost each year
Mobility	Enable operators and drivers to make informed decisions Reduce travel delays Communication between vehicles and traffic management optimize routing of vehicles
Environment	Reduces CO2 emissions by reducing travel times Provides real-time information

Benefits of connected roadways :

i. reduced traffic jams and urban congestion
ii. decreased casualties and fatalities
iii. increased response time for emergency vehicles
iv. reduced vehicle emissions

IoT applications in connected roadways-

a) Intersection Movement Assist (IMA) - warns a driver (or a self-driving car) when it is not safe to enter an intersection due to a high probability of a collision.
b) Automated vehicle tracking - a vehicle's location is used for notification of arrival times, theft prevention, or highway assistance.
c) Cargo management- provides precise positioning of cargo as it is enroute so that notification alerts can be sent to a dispatcher and routes can be optimized for congestion and weather.
d) Road weather communications- use sensors and data from satellites, roads, and bridges to warn vehicles of dangerous conditions or inclement weather on the current route.

A connected car is capable of generating continuous data related to location, performance, driver behavior, and much more, which will generate more than 25 GB of data per hour, which will be sent to the cloud.

Considering number of hours a car is driven per ,the number of cars on the road, and amount of data generated by connected car, transmitted, and stored in the cloud will be in the zettabytes per year.

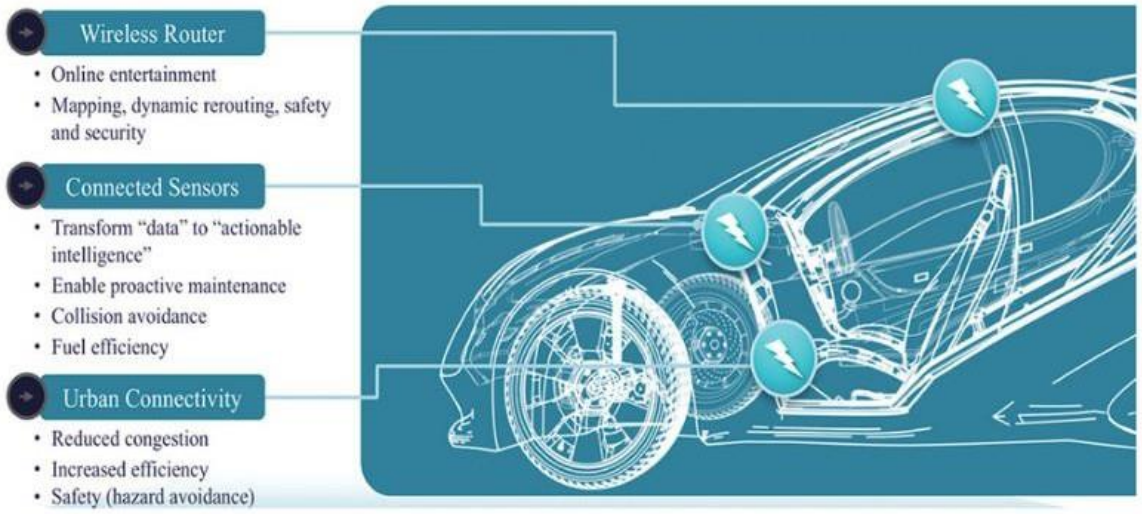


Figure 1-5 *The Connected Car*

- How the data generated by a car will be used by third parties?

❖ Tire companies can collect data related to use and durability of their products in real time.

❖ Automobile manufacturers collect to better understand how the cars are driven, when parts are starting to fail, or whether the car has broken down to build better cars in the future.

In the future, car sensors will be able to interact with third-party applications, such as GPS/maps, to enable dynamic rerouting to avoid traffic, accidents, and other hazards. Similarly, Internet-based entertainment, including music, movies, and other streamings or downloads, can be personalized and customized to optimize a road trip.

This data will also be used for targeted advertising. As GPS navigation systems become more integrated with sensors and wayfinding applications, it will become possible for personalized routing suggestions to be made. For example, if it is known that you prefer a certain coffee shop, through the use of a cloud-based data connector, the navigation system will be able to provide routing suggestions that have you drive your car past the right coffee shop.

Connected roadways are likely to be one of the biggest growth areas for innovation. Automobiles and the roads they use have seen incredible change over the past century, but the changes ahead of us are going to be just as astonishing. In the past few years alone, we have seen highway systems around the world adopt sophisticated sensors systems that can detect seismic vibrations, car accidents, severe weather conditions, traffic congestion, and more. Recent advancements in roadway fiber-optic sensing technology is now able to record not only how many cars are passing but their speed and type.

1.4.2 Connected Factory

The main challenges facing manufacturing in a factory:

- Accelerating new product and service introductions to meet customer and market opportunities
- Increasing plant production, quality, and uptime while decreasing cost
- Mitigating unplanned downtime (which wastes, on average, at least 5% of production)
- Securing factories from cyber threats
- Decreasing high cabling and re-cabling costs (up to 60% of deployment costs)
- Improving worker productivity and safety

A convergence of factory-based operational technologies and architectures with global IT networks is referred to as the **connected factory**.

Sensors communicate using the Internet Protocol (IP) over an Ethernet infrastructure. They transmit and receive large quantities of real-time informational and diagnostic data. More IP-enabled devices such as video cameras, diagnostic smart objects, and even personal mobile devices, are being added to the manufacturing environment.

For example, a smelting facility extracts metals from their ores. The facility uses both heat and chemicals to decompose the ore, leaving behind the base metal. This is a multistage process, and the data and controls are all accessed via various control rooms in a facility.

Example: real-time location system (RTLS).
An RTLS utilizes small and easily deployed Wi-Fi RFID tags that attach to virtually any material and provide real-time location and status. These tags enable a facility to track production as it happens. These IoT sensors allow components and materials on an assembly line to "talk" to the network. If each assembly line's output is tracked in real time, decisions can be made to speed up or slow production to meet targets, and it is easy to determine how quickly employees are completing the various stages of production. Bottlenecks at any point in production and quality problems are also quickly identified.

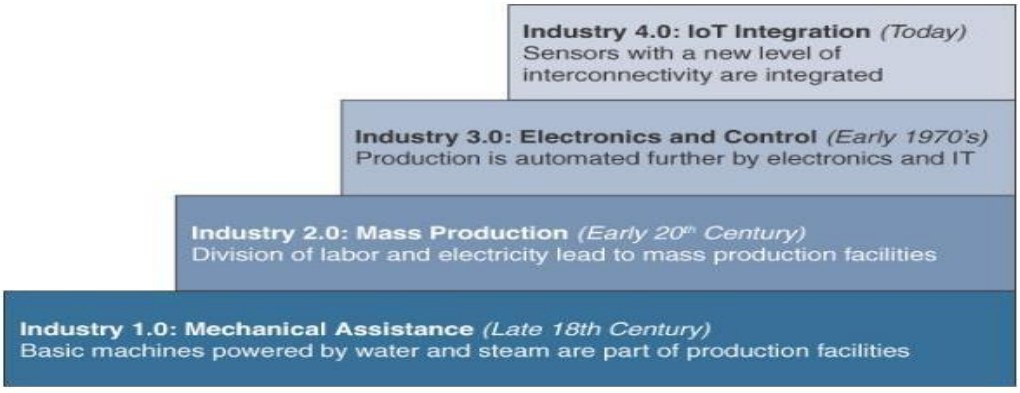


Figure 1-6 *The Four Industrial Revolutions*

The IoT wave of Industry 4.0 takes manufacturing from a purely automated assembly line model of production to a model where the machines are intelligent and communicate with one another. IoT in manufacturing brings with it the opportunity for inserting intelligence into factories. This starts with creating smart objects, which involves embedding sensors, actuators, and controllers into just about everything related to production.

1.4.3 Smart Connected Buildings

Another place IoT is making a disruptive impact is in the smart connected buildings space. In the past several decades, buildings have become increasingly complex, with systems overlaid one upon another, resulting in complex intersections of structural, mechanical, electrical, and IT components. Over time, these operational networks that support the building environment have matured into sophisticated systems; however, for the most part, they are deployed and managed as separate systems that have little to no interaction with each other.

The function of a building is to provide a work environment that keeps the workers comfortable, efficient, and safe. Work areas need to be well lit and kept at a comfortable temperature. To keep workers safe, the fire alarm and suppression system needs to be carefully managed, as do the door and physical security alarm systems.

Motion detection occupancy sensors work great if everyone is moving around in a crowded room and can automatically shut the lights off when everyone has left, but what if a person in the room is out of sight of the sensor? It is a frustrating matter to be at the mercy of an unintelligent sensor on the wall that wants to turn off the lights on you.

Similarly, sensors are often used to control the heating, ventilation, and air-conditioning (HVAC) system. Temperature sensors are spread throughout the building and are used to influence the building management system's (BMS's) control of air flow into a room.

Before you can bring together heterogeneous systems, they need to converge at the network layer and support a common services layer that allows application integration. For example, the de facto communication protocol responsible for building automation is known as BACnet (Building Automation and Control Network). In a nutshell, the BACnet protocol defines a set of services that allow Ethernet-based communication between building devices such as HVAC, lighting, access control, and fire detection systems. The same building Ethernet switches used for IT may also be used for BACnet. This standardization also makes possible an intersection point to the IP network (which is run by the IT department) through the use of a gateway device. In addition, BACnet/IP has been defined to allow the "things" in the building network to communicate over IP, thus allowing closer consolidation of the building management system on a single network. Figure 1-7 illustrates the conversion of building protocols to IP over time.

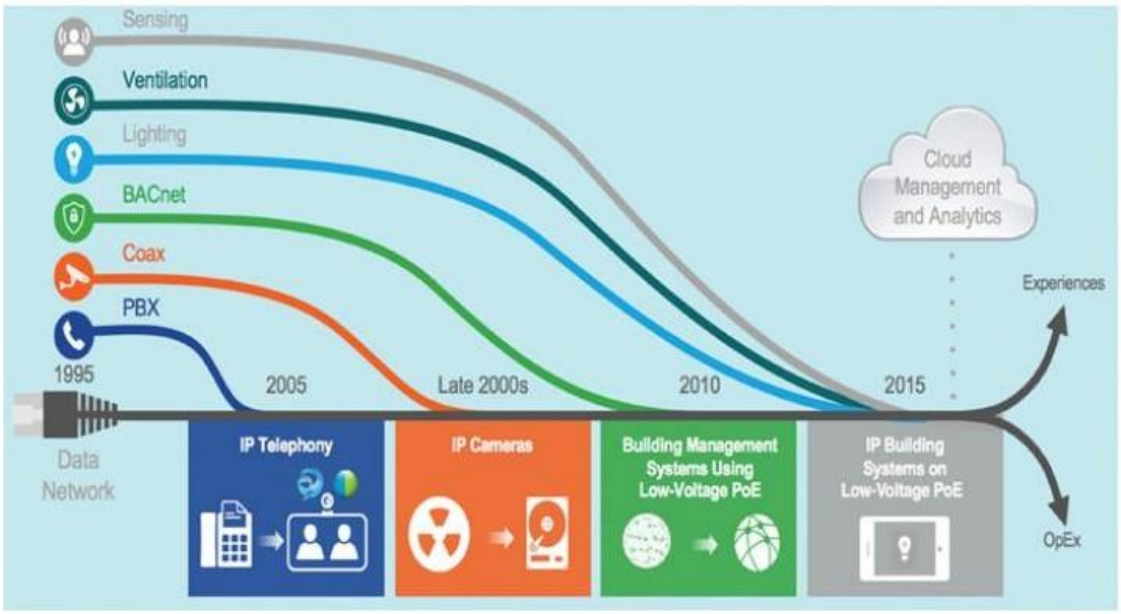


Figure 1-7 *Convergence of Building Technologies to IP*

Another promising IoT technology in the smart connected building, and one that is seeing widespread adoption, is the "digital ceiling." The digital ceiling is more than just a lighting control system. This technology encompasses several of the building's different networks—including lighting, HVAC, blinds, CCTV (closed-circuit television), and security systems—and combines them into a single IP network. Figure 1-8 provides a framework for the digital ceiling.

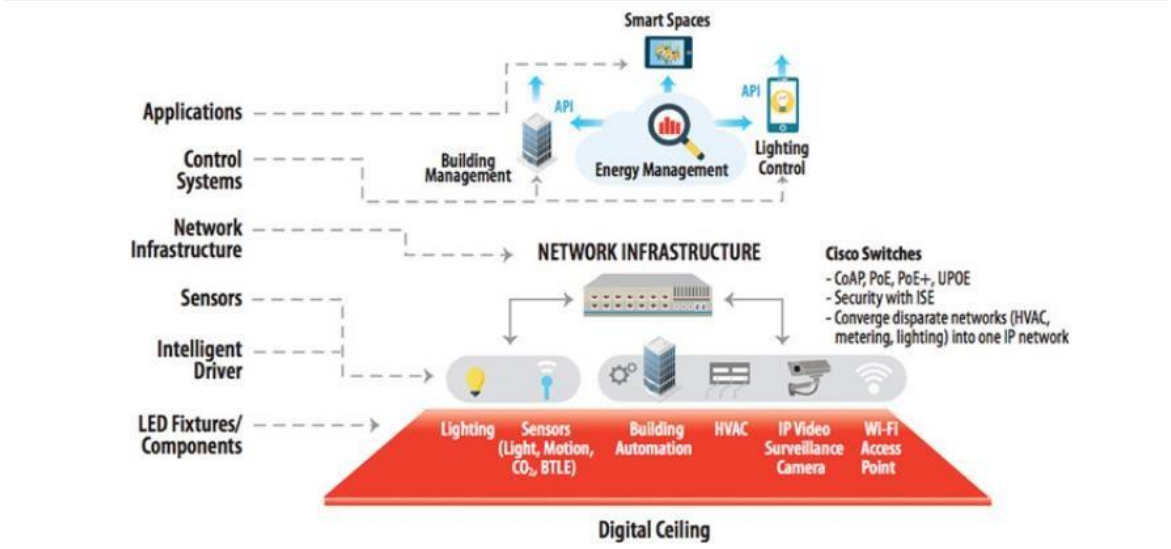


Figure 1-8 *A Framework for the Digital Ceiling*

Central to digital ceiling technology is the lighting system. As you are probably aware, the lighting market is currently going through a major shift toward light-emitting diodes (LEDs). Compared to traditional lighting, LEDs offer lower energy consumption and far longer life. The lower power requirements of LED fixtures allow them to run on Power over Ethernet (PoE), permitting them to be

connected to standard network switches.

In a digital ceiling environment, every luminaire or lighting fixture is directly network-attached, providing control and power over the same infrastructure. This transition to LED lighting means that a single converged network is now able to encompasses luminaires that are part of consolidated building management as well as elements managed by the IT network, supporting voice, video, and other data applications.

The energy savings value of PoE-enabled LED lighting in the ceiling is clear. However, having an IP-enabled sensor device in the ceiling at every point people may be present opens up an entirely new set of possibilities. For example, most modern LED ceiling fixtures support occupancy sensors. These sensors provide high-resolution occupancy data collection, which can be used to turn the lights on and off, and this same data can be combined with advanced analytics to control other systems, such as HVAC and security.

1.4.4 Smart Creatures

One of the most well-known applications of IoT with respect to animals focuses on what is often referred to as the "connected cow." Sparked, a Dutch company, developed a sensor that is placed in a cow's ear. The sensor monitors various health aspects of the cow as well as its location and transmits the data wirelessly for analysis by the farmer.

The data from each of these sensors is approximately 200 MB per year, and you obviously need a network infrastructure to make the connection with the sensors and store the information. Once the data is being collected, however, you get a complete view of the herd, with statistics on every cow. You can learn how environmental factors may be affecting the herd as a whole and about changes in diet. This enables early detection of disease as cows tend to eat less days before they show symptoms. These sensors even allow the detection of pregnancy in cows.

Another application of IoT to organisms involves the placement of sensors on roaches. Researchers at North Carolina State University are working with Madagascar hissing cockroaches in the hopes of helping emergency personnel rescue survivors after a disaster. An electronic backpack attaches to a roach. This backpack communicates with the roach through parts of its body. Low-level electrical pulses to an antenna on one side makes the roach turn to the opposite side because it believes it is encountering an obstacle. The cerci of the roach are sensory organs on the abdomen that detect danger through changing air currents. When the backpack stimulates the cerci, the roach moves forward because it thinks a predator is approaching.

1.5 CONVERGENCE OF IT AND OT

IT supports connections to the Internet along with related data and technology systems and is focused on the secure flow of data across an organization.

OT monitors and controls devices and processes on physical operational systems. These systems include assembly lines, utility distribution networks, production facilities, roadway systems, and many more.

Table 1-3 highlights some of the differences between IT and OT networks and their various challenges.

Criterion	Industrial OT Network	Enterprise IT Network
Operational focus	Keep the business operating 24x7	Manage the computers, data, and employee communication system in a secure way
Priorities	1. Availability 2. Integrity 3. Security	1. Security 2. Integrity 3. Availability
Types of data	Monitoring, control, and supervisory data	Voice, video, transactional, and bulk data
Security	Controlled physical access to devices	Devices and users authenticated to the network
Implication of failure	OT network disruption directly impacts business	Can be business impacting, depending on industry, but workarounds may be possible
Network upgrades (software or hardware)	Only during operational maintenance windows	Often requires an outage window when workers are not onsite; impact can be mitigated
Security vulnerability	Low: OT networks are isolated and often use proprietary protocols	High: continual patching of hosts is required, and the network is connected to Internet and requires vigilant protection

Criterion	Industrial OT Network	Enterprise IT Network
Operational focus	Keep the business operating 24x7	Manage the computers, data, and employee communication system in a secure way
Priorities	1. Availability 2. Integrity 3. Security	1. Security 2. Integrity 3. Availability
Types of data	Monitoring, control, and supervisory data	Voice, video, transactional, and bulk data
Security	Controlled physical access to devices	Devices and users authenticated to the network
Implication of failure	OT network disruption directly impacts business	Can be business impacting, depending on industry, but workarounds may be possible

Source: Maciej Kranz, *IT Is from Venus, OT Is from Mars*, blogs.cisco.com/digital/it-is-from-venus-ot-is-from-mars, July 14, 2015.

Table 1-3 *Comparing Operational Technology (OT) and Information Technology (IT)*

With the rise of IoT and standards-based protocols, such as IPv6, the IT and OT worlds are converging or, more accurately, OT is beginning to adopt the network protocols, technology, transport, and methods of the IT organization, and the IT organization is beginning to support the operational requirements used by OT. When IT and OT begin using the same networks, protocols, and processes, there are clear economies of scale. Not only does convergence reduce the amount of capital infrastructure needed but networks become easier to operate, and the flexibility of open standards allows faster growth and adaptability to new technologies.

With the merging of OT and IT, improvements are being made to both systems. OT is looking more toward IT technologies with open standards, such as Ethernet and IP. At the same time, IT is becoming more of a business partner with OT by better understanding business outcomes and operational requirements.

The overall benefit of IT and OT working together is a more efficient and profitable business due to reduced downtime, lower costs through economy of scale, reduced inventory, and improved delivery times. When IT/OT convergence is managed correctly, IoT becomes fully supported by both groups.

This provides a "best of both worlds" scenario, where solid industrial control systems reside on an open, integrated, and secure technology foundation.

1.6 IOT CHALLENGES

Challenge	Description
Scale	While the scale of IT networks can be large, the scale of OT can be several orders of magnitude larger. For example, one large electrical utility in Asia recently began deploying IPv6-based smart meters on its electrical grid. While this utility company has tens of thousands of employees (which can be considered IP nodes in the network), the number of meters in the service area is tens of millions. This means the scale of the network the utility is managing has increased by more than 1,000-fold! Chapter 5, "IP as the IoT Network Layer," explores how new design approaches are being developed to scale IPv6 networks into the millions of devices.
Security	With more "things" becoming connected with other "things" and people, security is an increasingly complex issue for IoT. Your threat surface is now greatly expanded, and if a device gets hacked, its connectivity is a major concern. A compromised device can serve as a launching point to attack other devices and systems. IoT security is also pervasive across just about every facet of IoT. For more information on IoT security, see Chapter 8, "Securing IoT."
Privacy	As sensors become more prolific in our everyday lives, much of the data they gather will be specific to individuals and their activities. This data can range from health information to shopping patterns and transactions at a retail establishment. For businesses, this data has monetary value. Organizations are now discussing who owns this data and how individuals can control whether it is shared and with whom.
Big data and data analytics	IoT and its large number of sensors is going to trigger a deluge of data that must be handled. This data will provide critical information and insights if it can be processed in an efficient manner. The challenge, however, is evaluating massive amounts of data arriving from different sources in various forms and doing so in a timely manner. See Chapter 7 for more information on IoT and the challenges it faces from a big data perspective.
Interoperability	As with any other nascent technology, various protocols and architectures are jockeying for market share and standardization within IoT. Some of these protocols and architectures are based on proprietary elements, and others are open. Recent IoT standards are helping minimize this problem, but there are often various protocols and implementations available for IoT networks. The prominent protocols and architectures—especially open, standards-based implementations—are the subject of this book. For more information on IoT architectures, see Chapter 2, "IoT Network Architecture and Design." Chapter 4, "Connecting Smart Objects," Chapter 5, "IP as the IoT Network Layer," and Chapter 6, "Application Protocols for IoT," take a more in-depth look at the protocols that make up IoT.

Table 1-4 *IoT Challenges*

1.7 DRIVERS BEHIND NEW NETWORK ARCHITECTURES

While the architect has extensive experience in designing homes, those skills will clearly not be enough to meet the demands of this new project. The scale of the stadium is several magnitudes larger, the use is completely different, and the wear and tear will be at a completely different level. The architect needs a new architectural approach that meets the requirements for building the stadium.

The key difference between IT and IoT is the data. While IT systems are mostly concerned with reliable and continuous support of business applications such as email, web, databases, CRM systems, and so on, IoT is all about the data generated by sensors and how that data is used. The essence of IoT architectures thus involves how the data is transported, collected, analyzed, and ultimately acted upon.

Table 2-1 takes a closer look at some of the differences between IT and IoT networks, with a focus on the IoT requirements that are driving new network architectures, and considers what adjustments are needed.

Challenge	Description	IoT Architectural Change Required
Scale	The massive scale of IoT end-points (sensors) is far beyond that of typical IT networks.	The IPv4 address space has reached exhaustion and is unable to meet IoT's scalability requirements. Scale can be met only by using IPv6. IT networks continue to use IPv4 through features like Network Address Translation (NAT).
Security	IoT devices, especially those on wireless sensor networks (WSNs), are often physically exposed to the world.	Security is required at every level of the IoT network. Every IoT endpoint node on the network must be part of the overall security strategy and must support device-level authentication and link encryption. It must also be easy to deploy with some type of a zero-touch deployment model.

Devices and networks constrained by power, CPU, memory, and link speed	Due to the massive scale and longer distances, the networks are often constrained, lossy, and capable of supporting only minimal data rates (tens of bps to hundreds of Kbps).	New last-mile wireless technologies are needed to support constrained IoT devices over long distances. The network is also constrained, meaning modifications need to be made to traditional network-layer transport mechanisms.
The massive volume of data generated	The sensors generate a massive amount of data on a daily basis, causing network bottlenecks and slow analytics in the cloud.	Data analytics capabilities need to be distributed throughout the IoT network, from the edge to the cloud. In traditional IT networks, analytics and applications typically run only in the cloud.
Support for legacy devices	An IoT network often comprises a collection of modern, IP-capable endpoints as well as legacy, non-IP devices that rely on serial or proprietary protocols.	Digital transformation is a long process that may take many years, and IoT networks need to support protocol translation and/or tunneling mechanisms to support legacy protocols over standards-based protocols, such as Ethernet and IP.
The need for data to be analyzed in real time	Whereas traditional IT networks perform scheduled batch processing of data, IoT data needs to be analyzed and responded to in real-time.	Analytics software needs to be positioned closer to the edge and should support real-time streaming analytics. Traditional IT analytics software (such as relational databases or even Hadoop), are better suited to batch-level analytics that occur after the fact.

Table 2-1 *IoT Architectural Drivers*

The following sections expand on the requirements driving specific architectural changes for IoT.

Scale

- The scale of a typical IT network is on the order of several thousand devices—typically printers, mobile wireless devices, laptops, servers, and so on.
- IoT introduces a model where an average-sized utility, factory, transportation system, or city could easily be asked to support a network of this scale.
- Based on scale requirements of this order, IPv6 is the natural foundation for the IoT network layer.

Security

- targeted malicious attacks using vulnerabilities in networked machines, such as the outbreak of the Stuxnet worm, which specifically affected Siemens programmable logic controller (PLC) systems.
- Protecting corporate data from intrusion and theft is one of the main functions of the IT department. IT departments go to great lengths to protect servers, applications, and the

network, setting up defense-in-depth models with layers of security designed to protect the cyber crown jewels of the corporation.

- IoT endpoints are often located in wireless sensor networks that use unlicensed spectrum and are not only visible to the world through a spectrum analyzer but often physically accessible and widely distributed in the field.

For optimum security, IoT systems must:

- Be able to identify and authenticate all entities involved in the IoT service
- Ensure that all user data shared between the endpoint device and back-end applications is encrypted
- Comply with local data protection legislation so that all data is protected and stored correctly
- Utilize an IoT connectivity management platform and establish rules-based security policies so immediate action can be taken if anomalous behavior is detected from connected devices
- Take a holistic, network-level approach to security

Constrained Devices and Networks

- Most IoT sensors have limited power, CPU, and memory, and they transmit only when there is something important.
- Because of the massive scale of these devices and the large, uncontrolled environments where they are usually deployed, the networks that provide connectivity also tend to be very lossy and support very low data rates.
- IoT requires a new breed of connectivity technologies that meet both the scale and constraint limitations.

Data

- In IoT the data is like gold, as it is what enables businesses to deliver new IoT services that enhance the customer experience, reduce cost, and deliver new revenue opportunities.
- Although most IoT-generated data is unstructured, the insights it provides through analytics can revolutionize processes and create new business models.
- IoT systems are designed to stagger data consumption throughout the architecture, both to filter and reduce unnecessary data going upstream and to provide the fastest possible response to devices when necessary.

Legacy Device Support

- In OT systems, end devices are likely to be on the network for a very long time—sometimes decades. As IoT networks are deployed, they need to support the older devices already present on the network, as well as devices with new capabilities.
- In many cases, legacy devices are so old that they don’t even support IP.
- IoT network must either be capable of some type of protocol translation or use a gatew
- ay device to connect these legacy endpoints to the IoT network.

1.9 COMPARING IOT ARCHITECTURES

The aforementioned challenges and requirements of IoT systems have driven a whole new discipline of network architecture. In the past several years, architectural standards and frameworks have emerged to address the challenge of designing massive-scale IoT networks.

The foundational concept in all these architectures is supporting data, process, and the functions that endpoint devices perform. Two of the best-known architectures are those supported by oneM2M and the IoT World Forum (IoTWF), discussed in the following sections.

The one M2M IoT Standardized Architecture

oneM2M's framework focuses on IoT services, applications, and platforms which include smart metering applications, smart grid, smart city automation, e-health, and connected vehicles.

One of the greatest challenges in designing an IoT architecture is dealing with the heterogeneity of devices, software, and access methods. By developing a horizontal platform architecture, oneM2M is developing standards that allow interoperability at all levels of the IoT stack. oneM2M's horizontal framework and RESTful APIs allow the LoRaWAN system to interface with the building management system over an IoT network, thus promoting end-to-end IoT communications in a consistent way, no matter how heterogeneous the networks.

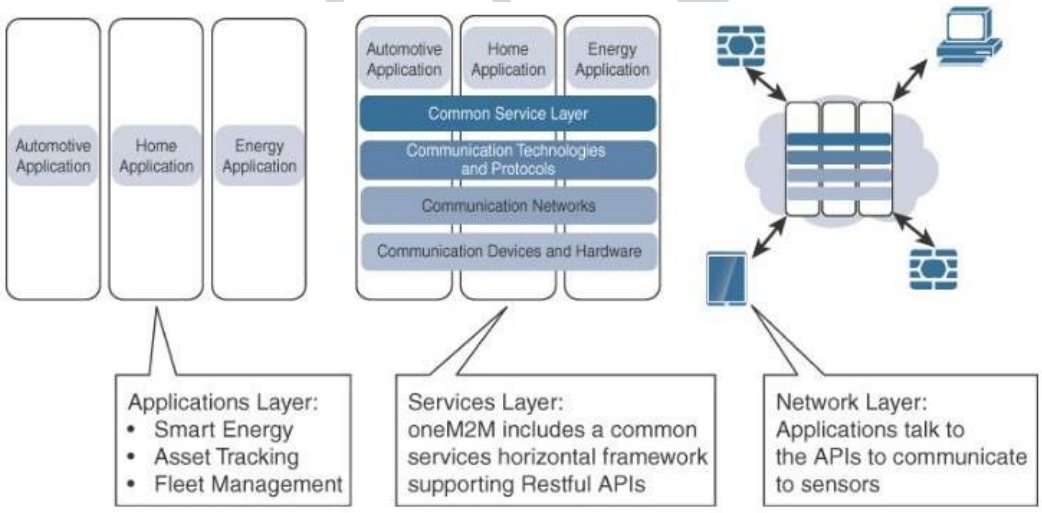


Figure 2-1 *The Main Elements of the oneM2M IoT Architecture*

Applications layer:

- It includes the application-layer protocols and attempts to standardize northbound API definitions for interaction with business intelligence (BI) systems.
- Applications tend to be industry-specific and have their own sets of data models, and thus they are shown as vertical entities.

Services layer:

- The horizontal modules include the physical network that the IoT applications run on, the underlying management protocols, and the hardware. Examples include backhaul communications via cellular, MPLS networks, VPNs, and so on.
- This conceptual layer adds APIs and middleware supporting third-party services and applications.
- One of the stated goals of oneM2M is to "develop technical specifications which address the need for a common M2M Service Layer that can be readily embedded within various hardware and software nodes, and rely upon connecting the myriad of devices in the field area network to M2M application servers, which typically reside in a cloud or data center."

Network layer:

- It includes the devices and the communications infrastructure which include wireless mesh technologies, such as IEEE 802.15.4, and wireless point-to-multipoint systems, such as IEEE 801.11ah.
- Also included are wired device connections, such as IEEE 1901 power line communications.
- The device domain includes the gateway device, which provides communications up into the core network and acts as a demarcation point between the device and network domains.

IoT World Forum (IoTWF) Standardized Architecture

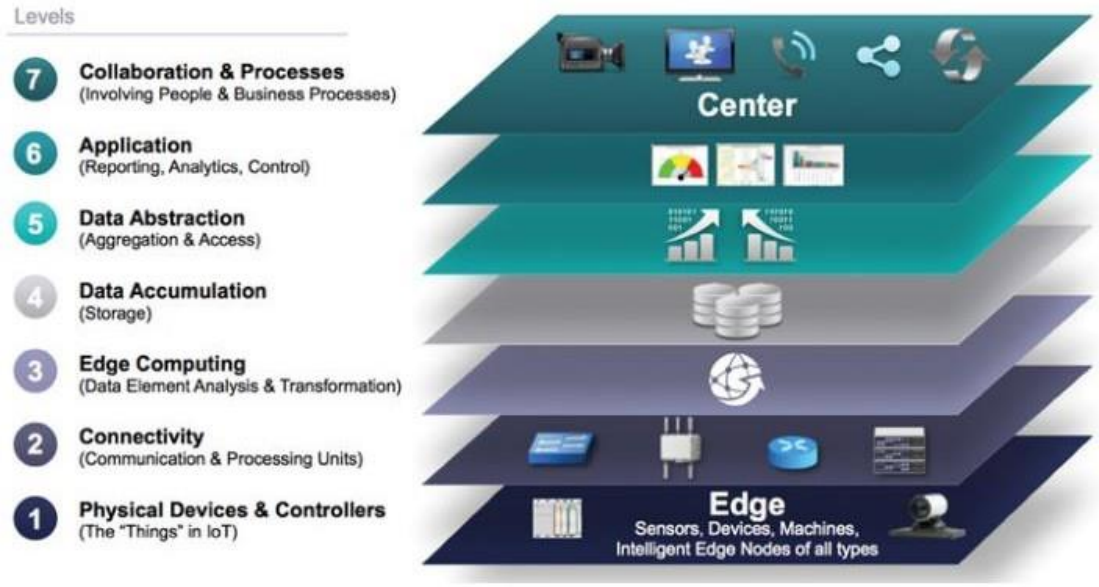


Figure 2-2 *IoT Reference Model Published by the IoT World Forum*

The seven layers of the IoT Reference Model:

Layer 1: Physical Devices and Controllers Layer

- This layer is home to the "things" in the Internet of Things, including the various endpoint devices and sensors that send and receive information.
- The primary function is generating data and being capable of being queried and/or controlled over a network.

Layer 2: Connectivity Layer

The primary function of this IoT layer is the reliable and timely transmission of data.

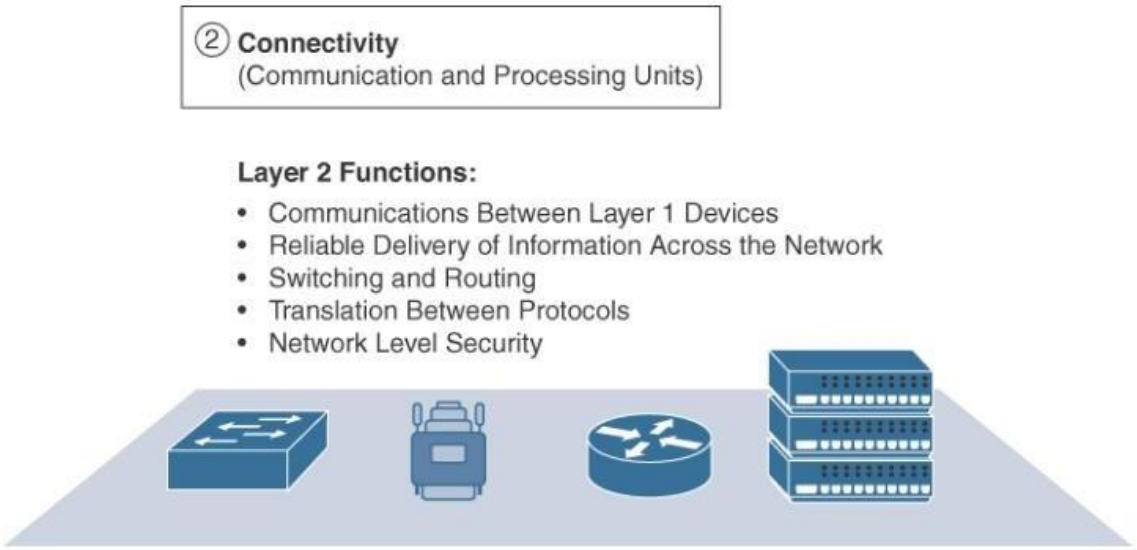


Figure 2-3 *IoT Reference Model Connectivity Layer Functions*

Layer 3: Edge Computing Layer

At this layer, the emphasis is on data reduction and converting network data flows into information that is ready for storage and processing by higher layers.

One of the basic principles of this reference model is that information processing is initiated as early

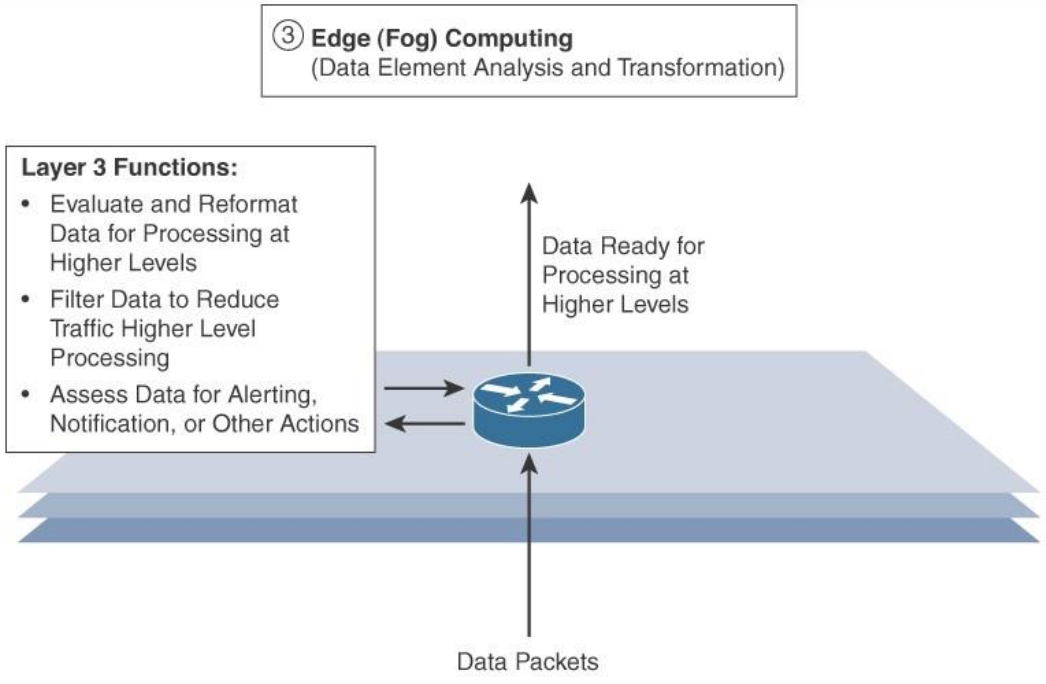


Figure 2-4 *IoT Reference Model Layer 3 Functions*

Upper Layers: Layers 4–7

IoT Reference Model Layer	Functions
Layer 4: Data accumulation layer	Captures data and stores it so it is usable by applications when necessary. Converts event-based data to query-based processing.
Layer 5: Data abstraction layer	Reconciles multiple data formats and ensures consistent semantics from various sources. Confirms that the data set is complete and consolidates data into one place or multiple data stores using virtualization.
Layer 6: Applications layer	Interprets data using software applications. Applications may monitor, control, and provide reports based on the analysis of the data.
Layer 7: Collaboration and processes layer	Consumes and shares the application information. Collaborating on and communicating IoT information often requires multiple steps, and it is what makes IoT useful. This layer can change business processes and delivers the benefits of IoT.

Table 2-2 *Summary of Layers 4–7 of the IoTWF Reference Model*

IT and OT Responsibilities in the IoT Reference Model

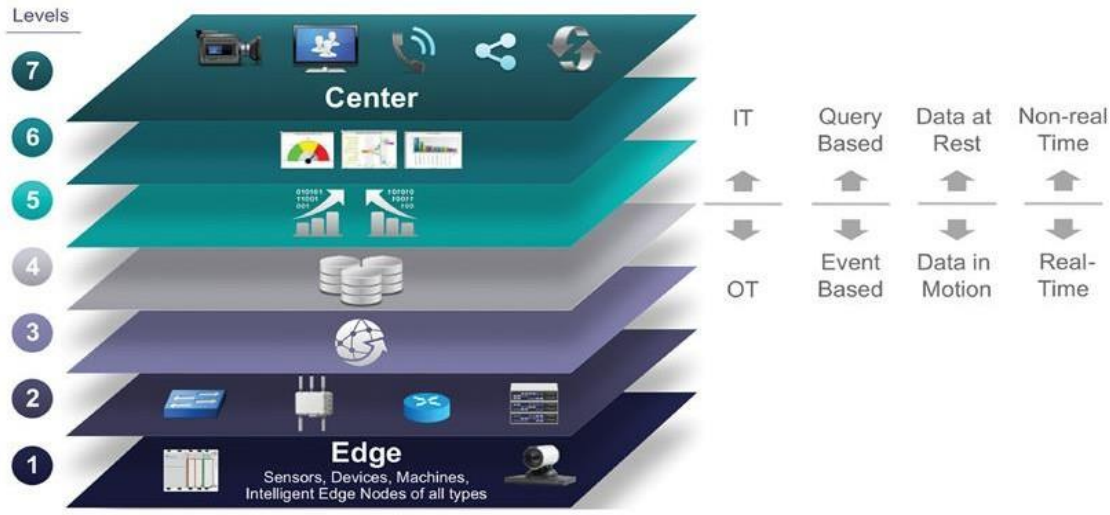


Figure 2-5 *IoT Reference Model Separation of IT and OT*

- The bottom of the stack is generally in the domain of OT. For an industry like oil and gas, this includes sensors and devices connected to pipelines, oil rigs, refinery machinery, and so on.
- The top of the stack is in the IT area and includes things like the servers, databases, and applications, all of which run on a part of the network controlled by IT.
- At the bottom, in the OT layers, the devices generate real-time data at their own rate—sometimes vast amounts on a daily basis.

Additional IoT Reference Models

IoT Reference Model	Description
Purdue Model for Control Hierarchy	The Purdue Model for Control Hierarchy (see www.cisco.com/c/en/us/td/docs/solutions/Verticals/EttF/EttFDIG/ch2_EttF.pdf) is a common and well-understood model that segments devices and equipment into hierarchical levels and functions. It is used as the basis for ISA-95 for control hierarchy, and in turn for the IEC-62443 (formerly ISA-99) cyber security standard. It has been used as a base for many IoT-related models and standards across industry. The Purdue Model's application to IoT is discussed in detail in Chapter 9, "Manufacturing," and in Chapter 10, "Oil & Gas."
Industrial Internet Reference Architecture (IIRA) by Industrial Internet Consortium (IIC)	The IIRA is a standards-based open architecture for Industrial Internet Systems (IISs). To maximize its value, the IIRA has broad industry applicability to drive interoperability, to map applicable technologies, and to guide technology and standard development. The description and representation of the architecture are generic and at a high level of abstraction to support the requisite broad industry applicability. The IIRA distills and abstracts common characteristics, features and patterns from use cases well understood at this time, predominantly those that have been defined in the IIC. For more information, see www.iiconsortium.org/IIRA.htm.
Internet of Things–Architecture (IoT-A)	IoT-A created an IoT architectural reference model and defined an initial set of key building blocks that are foundational in fostering the emerging Internet of Things. Using an experimental paradigm, IoT-A combined top-down reasoning about architectural principles and design guidelines with simulation and prototyping in exploring the technical consequences of architectural design choices. For more information, see https://vdivde-it.de/en.

1.10 A SIMPLIFIED IOT ARCHITECTURE

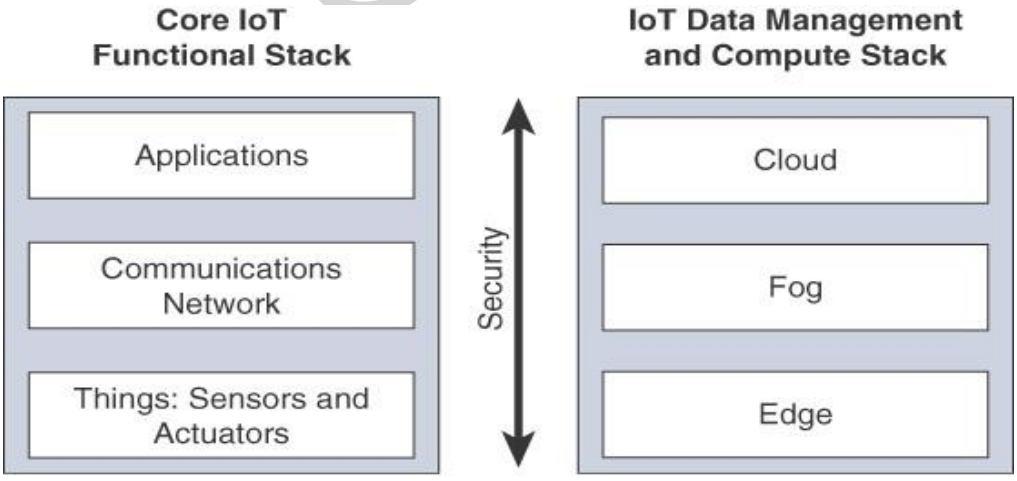


Figure 2-6 *Simplified IoT Architecture*

Nearly every published IoT model includes core layers including "things," a communications network, and applications. However, unlike other models, the framework presented here separates the core IoT and data management into parallel and aligned stacks, allowing you to carefully examine the functions of both the network and the applications at each stage of a complex IoT system. This separation gives you better visibility into the functions of each layer.

The presentation of the Core IoT Functional Stack in three layers is meant to simplify your understanding of the IoT architecture into its most foundational building blocks. Of course, such a simple architecture needs to be expanded on. The network communications layer of the IoT stack itself involves a significant amount of detail and incorporates a vast array of technologies. Consider for a moment the heterogeneity of IoT sensors and the many different ways that exist to connect them to a network. The network communications layer needs to consolidate these together, offer gateway and backhaul technologies, and ultimately bring the data back to a central location for analysis and processing.

Unlike with most IT networks, the applications and analytics layer of IoT doesn't necessarily exist only in the data center or in the cloud. Due to the unique challenges and requirements of IoT, it is often necessary to deploy applications and data management throughout the architecture in a tiered approach, allowing data collection, analytics, and intelligent controls at multiple points in the IoT system. In the model presented in this book, data management is aligned with each of the three layers of the Core IoT Functional Stack.

The three data management layers are the edge layer (data management within the sensors themselves), the fog layer (data management in the gateways and transit network), and the cloud layer (data management in the cloud or central data center).

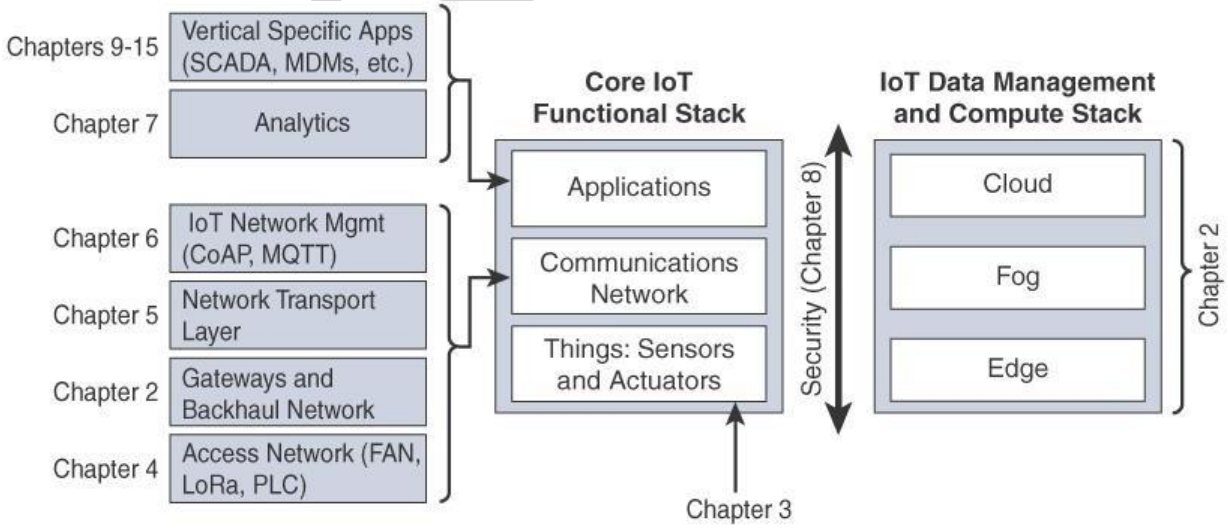


Figure 2-7 *Expanded View of the Simplified IoT Architecture*

The communications layer is broken down into four separate sub-layers: the access network, gateways and backhaul, IP transport, and operations and management sub-layers.

The applications layer of IoT networks is quite different from the application layer of a typical enterprise network. Instead of simply using business applications, IoT often involves a strong big data analytics component. One message that is stressed throughout this book is that IoT is not just about the control of IoT devices but, rather, the useful insights gained from the data generated by those devices. Thus, the applications layer typically has both analytics and industry-specific IoT control system components.

1.11 THE CORE IOT FUNCTIONAL STACK

IoT networks are built around the concept of "things," or smart objects performing functions and delivering new connected services. These objects are "smart" because they use a combination of contextual information and configured goals to perform actions.

From an architectural standpoint, several components have to work together for an IoT network to be operational:

A. **"Things" layer**: At this layer, the physical devices need to fit the constraints of the environment in which they are deployed while still being able to provide the information needed.
B. **Communications network layer**: When smart objects are not self-contained, they need to communicate with an external system. In many cases, this communication uses a wireless technology. This layer has four sublayers:
C. **Access network sublayer**: The last mile of the IoT network is the access network. This is typically made up of wireless technologies such as 802.11ah, 802.15.4g, and LoRa. The sensors connected to the access network may also be wired.
D. **Gateways and backhaul network sublayer**: A common communication system organizes multiple smart objects in a given area around a common gateway. The gateway communicates directly with the smart objects. The role of the gateway is to forward the collected information through a longer-range medium (called the backhaul) to a headend central station where the information is processed. This information exchange is a Layer 7 (application) function, which is the reason this object is called a gateway. On IP networks, this gateway also forwards packets from one IP network to another, and it therefore acts as a router.
E. **Network transport sublayer**: For communication to be successful, network and transport layer protocols such as IP and UDP must be implemented to support the variety of devices to connect and media to use.
F. **IoT network management sublayer**: Additional protocols must be in place to allow the headend applications to exchange data with the sensors. Examples include CoAP and MQTT.
G. **Application and analytics layer**: At the upper layer, an application needs to process the collected data, not only to control the smart objects when necessary, but to make intelligent decision based on the information collected and, in turn, instruct the "things" or other systems
H. to adapt to the analyzed conditions and change their behaviors or parameters.

The following sections examine these elements and help you architect your IoT communication network.

Layer 1: Things: Sensors and Actuators Layer

a) **Battery-powered or power-connected:** This classification is based on whether the object carries its own energy supply or receives continuous power from an external power source. Battery-powered things can be moved more easily than line-powered objects. However, batteries limit the lifetime and amount of energy that the object is allowed to consume, thus driving transmission range and frequency.

b) **Mobile or static:** This classification is based on whether the "thing" should move or always stay at the same location. A sensor may be mobile because it is moved from one object to another (for example, a viscosity sensor moved from batch to batch in a chemical plant) or because it is attached to a moving object (for example, a location sensor on moving goods in a warehouse or factory floor). The frequency of the movement may also vary, from occasional to permanent. The range of mobility (from a few inches to miles away) often drives the possible power source.

c) **Low or high reporting frequency:** This classification is based on how often the object should report monitored parameters. A rust sensor may report values once a month. A motion sensor may report acceleration several hundred times per second. Higher frequencies drive higher energy consumption, which may create constraints on the possible power source (and therefore the object mobility) and the transmission range.

d) **Simple or rich data:** This classification is based on the quantity of data exchanged at each report cycle. A humidity sensor in a field may report a simple daily index value (on a binary scale from 0 to 255), while an engine sensor may report hundreds of parameters, from temperature to pressure, gas velocity, compression speed, carbon index, and many others. Richer data typically drives higher power consumption. This classification is often combined with the previous to determine the object data throughput (low throughput to high throughput). You may want to keep in mind that throughput is a combined metric. A medium-throughput object may send simple data at rather high frequency (in which case the flow structure looks continuous), or may send rich data at rather low frequency (in which case the flow structure looks bursty).

e) **Report range:** This classification is based on the distance at which the gateway is located. For example, for your fitness band to communicate with your phone, it needs to be located a few meters away at most. The assumption is that your phone needs to be at visual distance for you to consult the reported data on the phone screen. If the phone is far away, you typically do not use it, and reporting data from the band to the phone is not necessary. By contrast, a moisture sensor in the asphalt of a road may need to communicate with its reader several hundred meters or even kilometers away.

f) **Object density per cell:** This classification is based on the number of smart objects (with a similar need to communicate) over a given area, connected to the same gateway. An oil pipeline may utilize a single sensor at key locations every few miles. By contrast, telescopes like the SETI Colossus telescope at the Whipple Observatory deploy hundreds, and sometimes thousands, of mirrors over a small area, each with multiple gyroscopes, gravity, and vibration sensors.

From a network architectural standpoint, your initial task is to determine which technology should be used to allow smart objects to communicate.

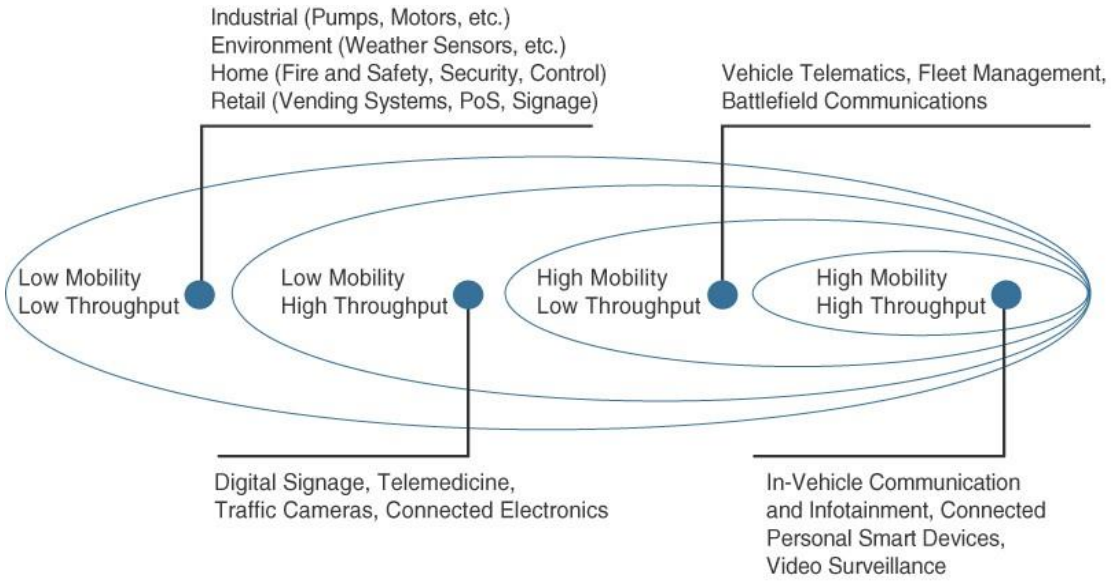


Figure 2-8 *Example of Sensor Applications Based on Mobility and Throughput*

Layer 2: Communications Network Layer

Once you have determined the influence of the smart object form factor over its transmission capabilities (transmission range, data volume and frequency, sensor density and mobility), you are ready to connect the object and communicate.

Compute and network assets used in IoT can be very different from those in IT environments. The difference in the physical form factors between devices used by IT and OT is obvious even to the most casual of observers. What typically drives this is the physical environment in which the devices are deployed. What may not be as inherently obvious, however, is their operational differences. The operational differences must be understood in order to apply the correct handling to secure the target assets.

Temperature variances are an easily understood metric. The cause for the variance is easily attributed to external weather forces and internal operating conditions. Remote external locations, such as those associated with mineral extraction or pipeline equipment can span from the heat to the cold .

- Humidity fluctuations can impact the long-term success of a system as well.
- Shock and vibration needs vary based on the deployment scenario.
- Solid particulates can also impact the gear. Most IT environments must contend with dust build-up that can become highly concentrated due to the effect of cooling fans.
- Hazardous location design may also cause corrosive impact to the equipment. Caustic materials can impact connections over which power or communications travel.
- Furthermore, they can result in reduced thermal efficiency by potentially coating the heat transfer surfaces.
- In some scenarios, the concern is not how the environment can impact the equipment but how the equipment can impact the environment.
- Power supplies in OT systems are also frequently different from those commonly seen on standard IT equipment. A wider range of power variations are common attributes of industrial compute components.

Access Network Sublayer

Range estimates are grouped by category names that illustrate the environment or the vertical where data collection over that range is expected. Common groups are as follows:

- PAN (personal area network): Scale of a few meters. This is the personal space around a person. A common wireless technology for this scale is Bluetooth.
- HAN (home area network): Scale of a few tens of meters. At this scale, common wireless technologies for IoT include ZigBee and Bluetooth Low Energy (BLE).
- NAN (neighborhood area network): Scale of a few hundreds of meters. The term NAN is often used to refer to a group of house units from which data is collected.

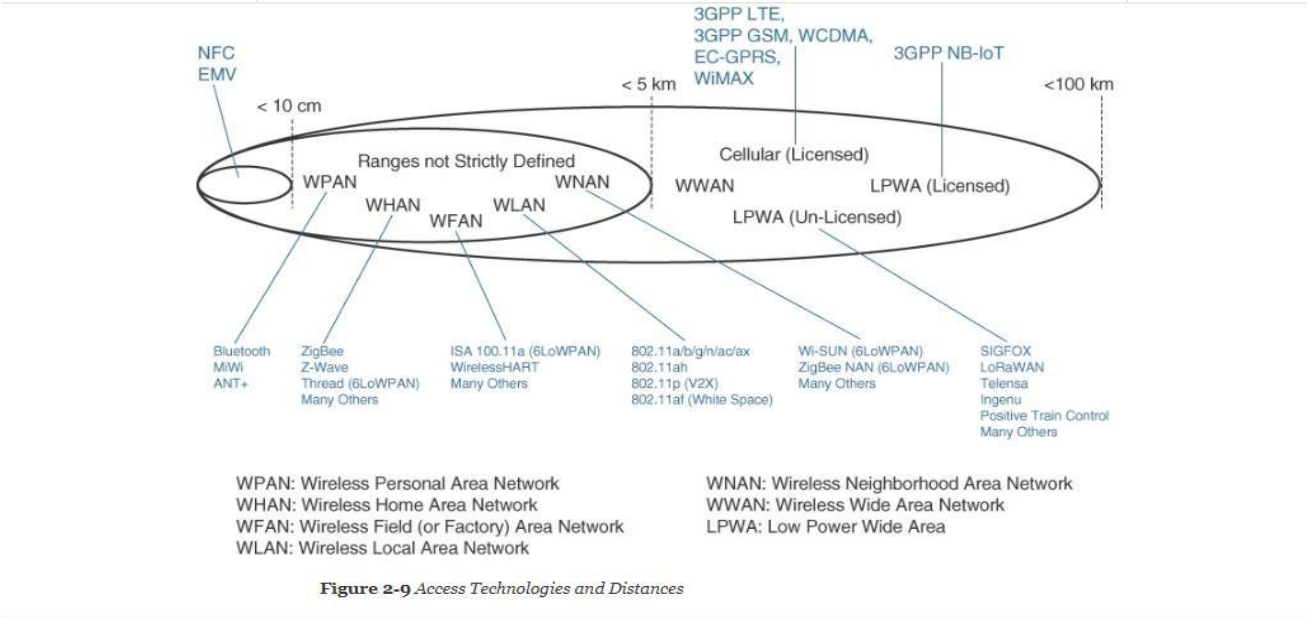


Figure 2-9 *Access Technologies and Distances*

- FAN (field area network): Scale of several tens of meters to several hundred meters. FAN typically refers to an outdoor area larger than a single group of house units. The FAN is often seen as "open space" (and therefore not secured and not controlled). A FAN is sometimes viewed as a group of NANs, but some verticals see the FAN as a group of HANs or a group of smaller outdoor cells.
- LAN (local area network): Scale of up to 100 m. This term is very common in networking, and it is therefore also commonly used in the IoT space when standard networking technologies (such as Ethernet or IEEE 802.11) are used.

Figure 2-11 combines cost, range, power consumption, and typical available bandwidth for common IoT access technologies.

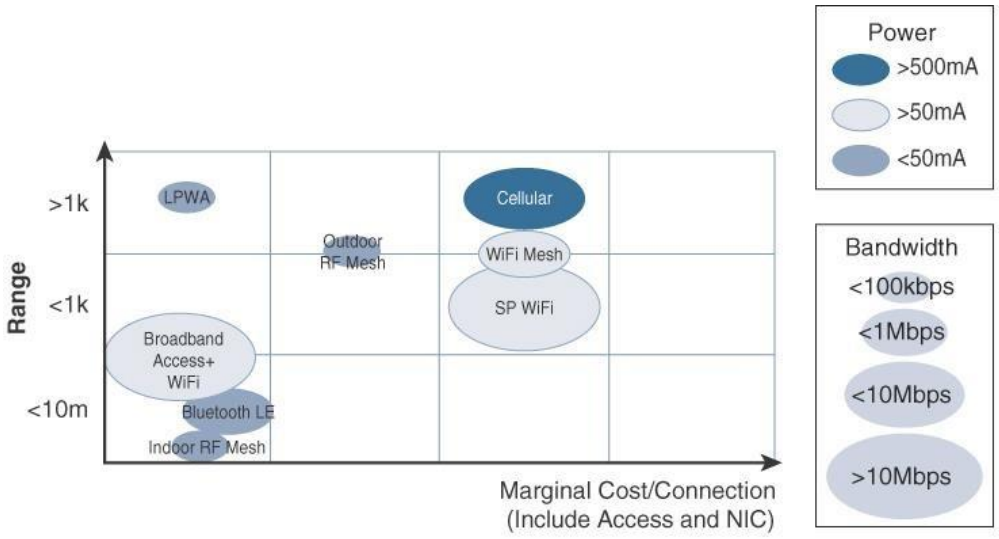


Figure 2-11 *Comparison Between Common Last-Mile Technologies in Terms of Range Versus Cost, Power, and Bandwidth*

The amount of data to carry over a given time period along with correlated power consumption (driving possible limitations in mobility and range) determines the wireless cell size and structure.

Similar ranges also do not mean similar topologies. Some technologies offer flexible connectivity structure to extend communication possibilities:

- **Point-to-point topologies**: These topologies allow one point to communicate with another point. This topology in its strictest sense is uncommon for IoT access, as it would imply that a single object can communicate only with a single gateway. However, several technologies are referred to as "point-to-point" when each object establishes an individual session with the gateway. The "point-to-point" concept, in that case, often refers to the communication structure more than the physical topology.
- **Point-to-multipoint topologies**: These topologies allow one point to communicate with more than one other point. Most IoT technologies where one or more than one gateways communicate with multiple smart objects are in this category. However, depending on the features available on each communicating mode, several subtypes need to be considered. A particularity of IoT networks is that some nodes (for example, sensors) support both data

collection and forwarding functions, while some other nodes (for example, some gateways) collect the smart object data, sometimes instruct the sensor to perform specific operations, and also interface with other networks or possibly other gateways. For this reason, some technologies categorize the nodes based on the functions (described by a protocol) they implement.

To form a network, a device needs to connect with another device. When both devices fully implement the protocol stack functions, they can form a peer-to-peer network. However, in many cases, one of the devices collects data from the others. For example, in a house, temperature sensors may be deployed in each room or each zone of the house, and they may communicate with a central point where temperature is displayed and controlled.
The sensor can implement a subset of protocol functions to perform just a specialized part (communication with the coordinator). Such a device is called a reduced-function device (RFD). An RFD cannot be a coordinator. An RFD also cannot implement direct communications to another RFD.

The coordinator that implements the full network functions is called, by contrast, a full-function device (FFD). An FFD can communicate directly with another FFD or with more than one FFD, forming multiple peer-to-peer connections. Topologies where each FFD has a unique path to another FFD are called cluster tree topologies. FFDs in the cluster tree may have RFDs, resulting in a cluster star topology.

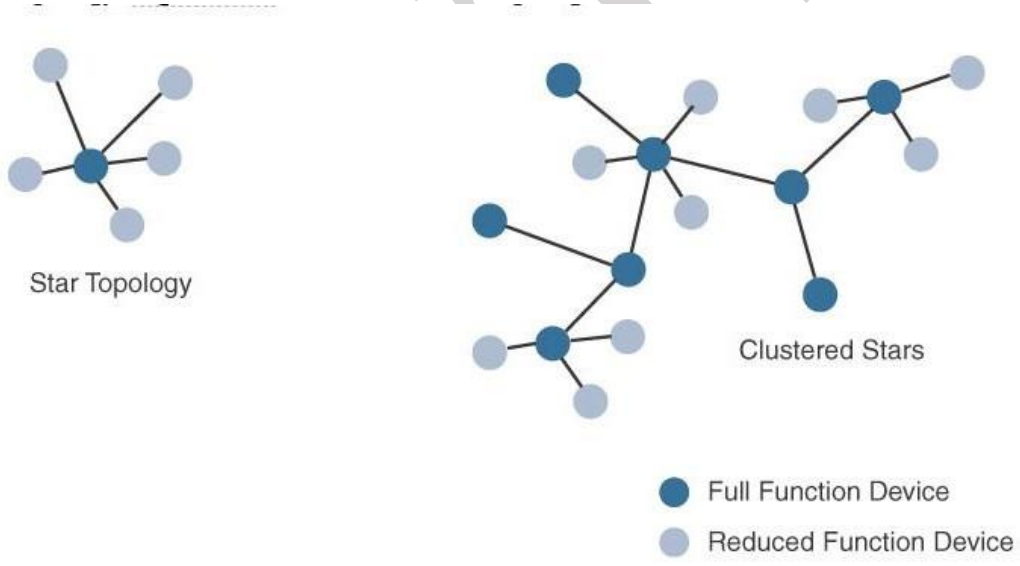


Figure 2-12 *Star and Clustered Star Topologies*

Point-to-multipoint technologies allow a node to have more than one path to another node, forming a mesh topology. This redundancy means that each node can communicate with more than just one other node. This communication can be used to directly exchange information between nodes (the receiver directly consumes the information received) or to extend the range of the communication link. In this case, an intermediate node acts as a relay between two other nodes. These two other

nodes would not be able to communicate successfully directly while respecting the constraints of power and modulation dictated by the PHY layer protocol.
Another property of mesh networks is redundancy. The disappearance of one node does not necessarily interrupt network communications. Nodes A and D are too far apart to communicate directly. In this case, communication can be relayed through nodes B or C. Node B may be used as the primary relay. However, the loss of node B does not prevent the communication between nodes A and D. Here, communication is rerouted through another node, node C.

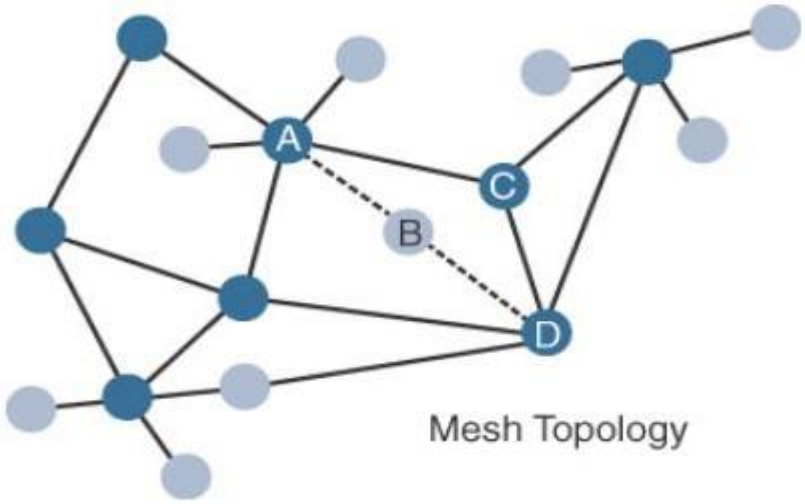


Figure 2-13 *Mesh Topology*

Gateways and Backhaul Sublayer

Data collected from a smart object may need to be forwarded to a central station where data is processed. As this station is often in a different location from the smart object, data directly received from the sensor through an access technology needs to be forwarded to another medium (the backhaul) and transported to the central station. The gateway is in charge of this inter-medium communication.

In the DSRC case, the entire "sensor field" is moving along with the gateway, but the general principles of IoT networking remain the same. The range at which DSRC can communicate is limited. Similarly, for all other IoT architectures, the choice of a backhaul technology depends on the communication distance and also on the amount of data that needs to be forwarded. When the smart object's operation is controlled from a local site, and when the environment is stable (for example, factory or oil and gas field), Ethernet can be used as a backhaul. Mesh is a common topology to allow communication flexibility in this type of dynamic environment.

Table 2-4 compares the main solutions from an architectural angle.

Technology	Type and Range	Architectural Characteristics
Ethernet	Wired, 100 m max	Requires a cable per sensor/sensor group; adapted to static sensor position in a stable environment; range is limited; link is very reliable
Wi-Fi (2.4 GHz, 5 GHz)	Wireless, 100 m (multipoint) to a few kilometers (P2P)	Can connect multiple clients (typically fewer than 200) to a single AP; range is limited; adapted to cases where client power is not an issue (continuous power or client battery recharged easily); large bandwidth available, but interference from other systems likely; AP needs a cable
802.11ah (HaloW, Wi-Fi in sub-1 GHz)	Wireless, 1.5 km (multipoint), 10 km (P2P)	Can connect a large number of clients (up to 6000 per AP); longer range than traditional Wi-Fi; power efficient; limited bandwidth; low adoption; and cost may be an issue
WiMAX (802.16)	Wireless, several kilometers (last mile), up to 50 km (backhaul)	Can connect a large number of clients; large bandwidth available in licensed spectrum (fee-based); reduced bandwidth in license-free spectrum (interferences from other systems likely); adoption varies on location
Cellular (for example, LTE)	Wireless, several kilometers	Can connect a large number of clients; large bandwidth available; licensed spectrum (interference-free; license-based)

Table 2-4 *Architectural Considerations for WiMAX and Cellular Technologies*

Network Transport Sub-

Distribution automation (DA) allows your meter to communicate with neighboring meters or other devices in the electrical distribution grid. With such communication, consumption load balancing may be optimized. For example, your air conditioning pulses fresh air at regular intervals. With DA, your neighbor's AC starts pulsing when your system pauses; in this way, the air in both houses is kept fresh, but the energy consumed from the network is stable instead of spiking up and down with uncoordinated start and stop points.

Similarly, your smart meter may communicate with your house appliances to evaluate their type and energy demand. With this scheme, your washing machine can be turned on in times of lower consumption from other systems, such as at night, while power to your home theater system will never be deprived, always turning on when you need it. Once the system learns your consumption pattern, charging of your electric car can start and stop at intervals to achieve the same overnight charge without creating spikes in energy demand. Data may flow locally, or it may have to be orchestrated by a central application to coordinate the power budget between houses.

This communication structure thus may involve peer-to-peer, point-to-point, point-to-multipoint, unicast and multicast communications. In a multitenant environment (for example, electricity and gas consumption management), different systems may use the same communication pathways. This communication occurs over multiple media (for example, power lines inside your house or a short-range wireless system like indoor Wi-Fi and/or ZigBee), a longer-range wireless system to the gateway, and yet another wireless or wired medium for backhaul transmission.

To allow for such communication structure, a network protocol with specific characteristics needs to be implemented. The protocol needs to be open and standard-based to accommodate multiple industries and multiple media. Scalability (to accommodate thousands or millions of sensors in a single network) and security are also common requirements. IP is a protocol that matches all these requirements. The advantages of IP are covered in depth in Chapter 5.

The flexibility of IP allows this protocol to be embedded in objects of very different natures, exchanging information over very different media, including low-power, lossy, and low-bandwidth networks. For example, RFC 2464 describes how an IPv6 packet gets encapsulated over an Ethernet frame and is also used for IEEE 802.11 Wi-Fi. Similarly, the IETF 6LoWPAN working group specifies how IPv6 packets are carried efficiently over lossy networks, forming an "adaption layer" for IPv6, primarily for IoT networks.

Finally, the transport layer protocols built above IP (UDP and TCP) can easily be leveraged to decide whether the network should control the data packet delivery (with TCP) or whether the control task should be left to the application (UDP). UDP is a much lighter and faster protocol than TCP. However, it does not guarantee packet delivery. Both TCP and UDP can be secured with TLS/SSL (TCP) or DTLS (UDP). Chapter 6 takes a closer look at TCP and UDP for IoT networks.

IoT Network Management Sub-layer

IP, TCP, and UDP bring connectivity to IoT networks. Upper-layer protocols need to take care of data transmission between the smart objects and other systems. Multiple protocols have been leveraged or created to solve IoT data communication problems. Some networks rely on a push model (that is, a sensor reports at a regular interval or based on a local trigger), whereas others rely on a pull model (that is, an application queries the sensor over the network), and multiple hybrid approaches are also possible.

Following the IP logic, some IoT implementers have suggested HTTP for the data transfer phase. After all, HTTP has a client and server component. The sensor could use the client part to establish a connection to the IoT central application (the server), and then data can be exchanged. You can find HTTP in some IoT applications, but HTTP is something of a fat protocol and was not designed to operate in constrained environments with low memory, low power, low bandwidth, and a high rate of packet failure. Despite these limitations, other web-derived protocols have been suggested for the IoT space. One example is WebSocket. WebSocket is part of the HTML5 specification, and provides a simple bidirectional connection over a single connection. Some IoT solutions use WebSocket to

manage the connection between the smart object and an external application. WebSocket is often combined with other protocols, such as MQTT (described shortly) to handle the IoT-specific part of the communication.

To respond to the limits of web-based protocols, another protocol was created by the IETF Constrained Restful Environments (CoRE) working group: Constrained Application Protocol (CoAP). CoAP uses some methods similar to those of HTTP (such as Get, Post, Put, and Delete) but implements a shorter list, thus limiting the size of the header. CoAP also runs on UDP (whereas HTTP typically uses TCP). CoAP also adds a feature that is lacking in HTTP and very useful for IoT: observation. Observation allows the streaming of state changes as they occur, without requiring the receiver to query for these changes.

Layer 3: Applications and Analytics Layer

Analytics Versus Control Applications

A. Analytics application:
   - collects data from multiple smart objects
   - processes the collected data
   - displays information resulting from the data that was processed
   - application processes the data to convey a view of the network that cannot be obtained from solely looking at the information displayed by a single smart object.

B. Control application:
   - controls the behavior of the smart object or the behavior of an object related to the smart object.
   - used for controlling complex aspects of an IoT network with a logic that cannot be programmed inside a single IoT object

An example of control system architecture is SCADA. SCADA was developed as a universal method to access remote systems and send instructions. One example where SCADA is widely used is in the control and monitoring of remote terminal units (RTUs) on the electrical distribution grid.

Data Versus Network Analytics

A. Data analytics:
   - processes the data collected by smart objects and combines it to provide an intelligent view related to the IoT system.
   - data processing can be very complex and may combine multiple changing values over complex algorithms.
   - Data analytics can also monitor the IoT system itself. For example, a machine or robot in a factory can report data about its own movements.

B. Network analytics:
   - A loss of connectivity may result in an accident or degradation of operations efficiency.
   - the control module cannot modify local object behaviors anymore.

Data Analytics Versus Business Benefits

A smarter architectural choice may be to allow for an open system where the network is engineered to be flexible enough that other sensors may be added in the future, and where both upstream and downstream operations are allowed. This flexibility allows for additional processing of the existing sensors and also deeper and more efficient interaction with the connected objects. This enhanced data processing can result in new added value for businesses that are not envisioned at the time when the system is initially deployed.

An example of a flexible analytics and control application is Cisco Jasper, which provides a turnkey cloud-based platform for IoT management and monetization. Consider the case of vending machines deployed throughout a city. At a basic level, these machines can be connected, and sensors can be deployed to report when a machine is in an error state. A repair person can be sent to address the issue when such a state is identified. This type of alert is a time saver and avoids the need for the repair team to tour all the machines in turn when only one may be malfunctioning.

This alert system may also avoid delay between the time when a machine goes into the error state and the time when a repair team visits the machine location. With a static platform, this use case is limited to this type of alert. With a flexible platform like Cisco Jasper, new applications may be imagined and developed over time. For example, the machine sensors can be improved to also report when an item is sold. The central application can then be enhanced to process this information and analyze what item is most sold, in what location, at what times. This new view of the machines may allow for an optimization of the items to sell in machines in a given area. Systems may be implemented to adapt the goods to time, season, or location—or many other parameters that may have been analyzed. In short, architecting open systems opens the possibility for new applications.

Smart Services

Fundamentally, smart services use IoT and aim for efficiency. For example, sensors can be installed on equipment to ensure ongoing conformance with regulations or safety requirements. This angle of efficiency can take multiple forms, from presence sensors in hazardous areas to weight threshold violation detectors on trucks.

Smart services can also be used to measure the efficiency of machines by detecting machine output, speed, or other forms of usage evaluation. Entire operations can be optimized with IoT. In hospitality, for example, presence and motion sensors can evaluate the number of guests in a lobby and redirect personnel accordingly. The same type of action can be taken in a store where a customer is detected as staying longer than the typical amount of time in front of a shelf. Personnel can be

deployed to provide assistance. Movement of people and objects on factory floors can be analyzed to optimize the production flow.

Smart services can be integrated into an IoT system. For example, sensors can be integrated in a light bulb. A sensor can turn a light on or off based on the presence of a human in the room. An even smarter system can communicate with other systems in the house, learn the human movement pattern, and anticipate the presence of a human, turning on the light just before the person enters the room. An even smarter system can use smarter sensors that analyze multiple parameters to detect human mood and modify accordingly the light color to adapt to the learned preferences, or to convey either a more relaxing or a more dynamic environment.

Light bulbs are a simple example. By connecting to other systems in the house, efficiencies can be coordinated. For example, the house entry alarm system or the heating system can coordinate with the presence detector in a light bulb to adapt to detected changes. The alarm system can disable volumetric movement alarms in zones where a known person is detected. The heating system can adapt the temperature to human presence or detected personal preferences.

Similar efficiency can be extended to larger systems than a house. For example, smart grid applications can coordinate the energy consumption between houses to regulate the energy demand from the grid. We already mentioned that your washing machine may be turned on at night when the energy demand for heating and cooling is lower. Just as your air conditioning pulses can be coordinated with your neighbor's, your washing machine cycles can be coordinated with the appliances in your house and in the neighborhood to smooth the energy demand spikes on the grid.

Efficiency also applies to M2M communications. In mining environments, vehicles can communicate to regulate the flows between drills, draglines, bulldozers, and dump trucks, for example, making sure that a dump truck is always available when a bulldozer needs it. In smart cities, vehicles communicate. A traffic jam is detected and anticipated automatically by public transportation, and the system can temporarily reroute buses or regulate the number of buses servicing a specific line based on traffic and customer quantity, instantaneous or learned over trending.

1.12 IOT DATA MANAGEMENT AND COMPUTE STACK

One of the key messages in the first two chapters of this book is that the massive scale of IoT networks is fundamentally driving new architectures. For instance, Figure 1-2 in Chapter 1 illustrates how the "things" connected to the Internet are continuing to grow exponentially, with a prediction by Cisco that by 2020 there will be more than 50 billion devices connected to some form of an IP network. Clearly, traditional IT networks are not prepared for this magnitude of network devices. However, beyond the network architecture itself, consider the data that is generated by these devices. If the number of devices is beyond conventional numbers, surely the data generated by these devices must also be of serious concern.

In fact, the data generated by IoT sensors is one of the single biggest challenges in building an IoT system. In the case of modern IT networks, the data sourced by a computer or server is typically generated by the client/server communications model, and it serves the needs of the application. In sensor networks, the vast majority of data generated is unstructured and of very little use on its own. For example, the majority of data generated by a smart meter is nothing more than polling data; the communications system simply determines whether a network connection to the meter is still active. This data on its own is of very little value. The real value of a smart meter is the metering data read by the meter management system (MMS). However, if you look at the raw polling data from a different perspective, the information can be very useful. For example, a utility may have millions of meters covering its entire service area. If whole sections of the smart grid start to show an interruption of connectivity to the meters, this data can be analyzed and combined with other sources of data, such as weather reports and electrical demand in the grid, to provide a complete picture of what is happening. This information can help determine whether the loss of connection to the meters is truly a loss of power or whether some other problem has developed in the grid. Moreover, analytics of this data can help the utility quickly determine the extent of the service outage and repair the disruption in a timely fashion.

In most cases, the processing location is outside the smart object. A natural location for this processing activity is the cloud. Smart objects need to connect to the cloud, and data processing is centralized. However, this model also has limitations. As data volume, the variety of objects connecting to the network, and the need for more efficiency increase. These new requirements include the following:

- *Minimizing latency*: Analyzing data close to the device that collected the data can make a difference between averting disaster and a cascading system failure.
- *Conserving network bandwidth*: It is not practical to transport vast amounts of data from thousands or hundreds of thousands of edge devices to the cloud. Nor is it necessary because many critical analyses do not require cloud-scale processing and storage.
- *Increasing local efficiency*: Collecting and securing data across a wide geographic area with different environmental conditions may not be useful.

The volume of data also introduces questions about bandwidth management. As the massive amount of IoT data begins to funnel into the data center, does the network have the capacity to sustain this volume of traffic? Does the application server have the ability to ingest, store, and analyze the vast quantity of data that is coming in? This is sometimes referred to as the “impedance mismatch” of the data generated by the IoT system and the management application’s ability to deal with that data.

As illustrated in Figure 2-14, data management in traditional IT systems is very simple. The endpoints (laptops, printers, IP phones, and so on) communicate over an IP core network to servers in the data center or cloud. Data is generally stored in the data center, and the physical links from access to core are typically high bandwidth, meaning access to IT data is quick.

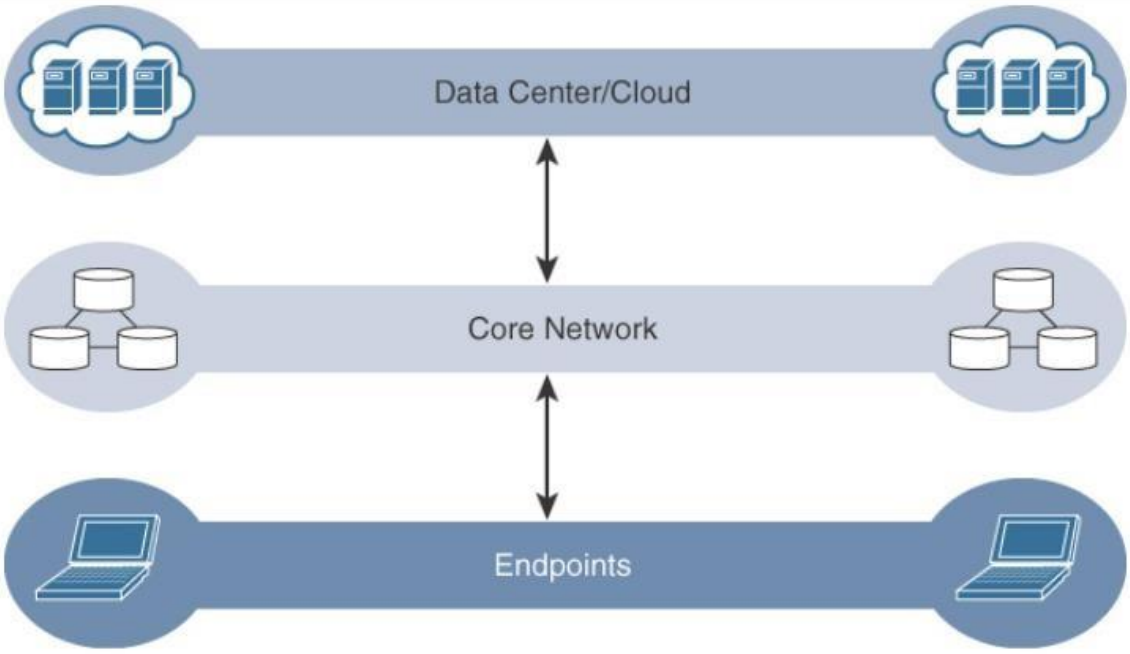


Figure 2-14 *The Traditional IT Cloud Computing Model*

IoT systems function differently. Several data-related problems need to be addressed:

a) Bandwidth in last-mile IoT networks is very limited. When dealing with thousands/millions of devices, available bandwidth may be on order of tens of Kbps per device or even less.
b) Latency can be very high. Instead of dealing with latency in the milliseconds range, large IoT networks often introduce latency of hundreds to thousands of milliseconds.
c) Network backhaul from the gateway can be unreliable and often depends on 3G/LTE or even satellite links. Backhaul links can also be expensive if a per-byte data usage model is necessary.
d) The volume of data transmitted over the backhaul can be high, and much of the data may not really be that interesting (such as simple polling messages).
e) Big data is getting bigger. The concept of storing and analyzing all sensor data in the cloud is impractical. The sheer volume of data generated makes real-time analysis and response to the data almost impossible.

Fog Computing

The solution to the challenges mentioned in the previous section is to distribute data management throughout the IoT system, as close to the edge of the IP network as possible. The best-known embodiment of edge services in IoT is fog computing.

Any device with computing, storage, and network connectivity can be a fog node. Examples include industrial controllers, switches, routers, embedded servers, and IoT gateways. Analyzing IoT data close to where it is collected minimizes latency, offloads gigabytes of network traffic from the core network, and keeps sensitive data inside the local network.

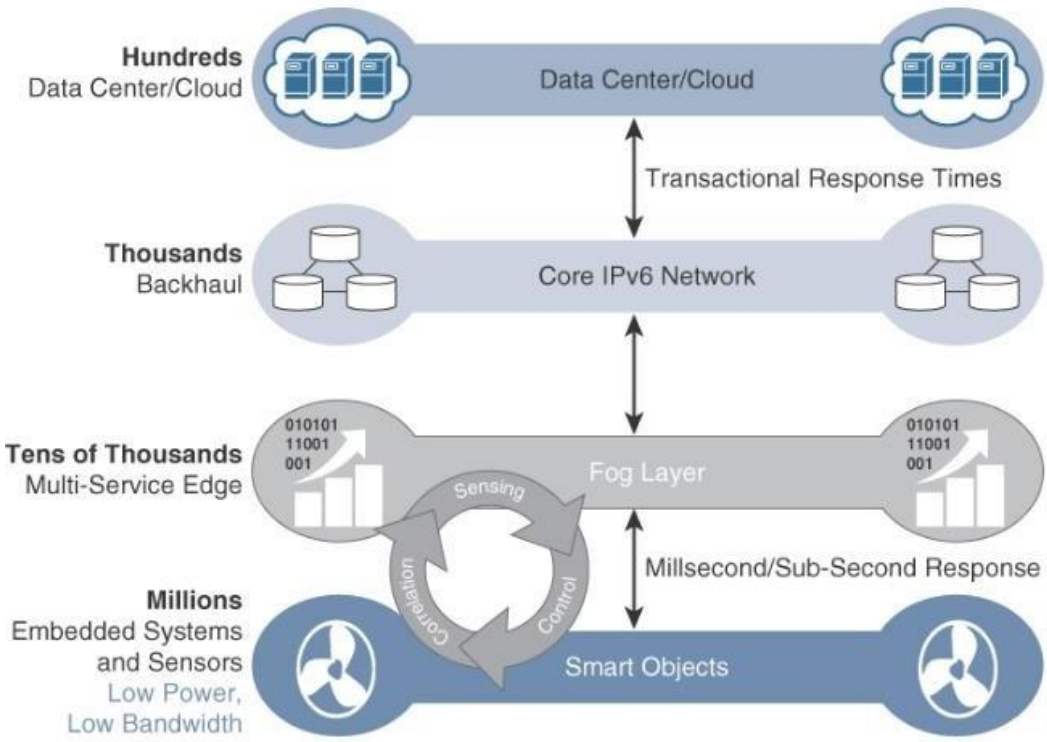


Figure 2-15 *The IoT Data Management and Compute Stack with Fog Computing*

i. Fog services are typically accomplished very close to the edge device, sitting as close to the IoT endpoints as possible.
ii. The fog node has contextual awareness of the sensors it is managing because of its geographic proximity to those sensors.
iii. The fog node is able to analyze information from all the sensors and can provide contextual analysis of the messages it is receiving and may decide to send back only the relevant information over the backhaul network to the cloud.

IoT fog computing enables data to be preprocessed and correlated with other inputs to produce relevant information. This data can then be used as real-time, actionable knowledge by IoT-enabled applications. Longer term, this data can be used to gain a deeper understanding of network behavior and systems for the purpose of developing proactive policies, processes, and responses.

Fog applications are as diverse as the Internet of Things itself. What they have in common is data reduction—monitoring or analyzing real-time data from network-connected things and then initiating an action, such as locking a door, changing equipment settings, applying the brakes on a train, zooming a video camera, opening a valve in response to a pressure reading, creating a bar chart, or sending an alert to a technician to make a preventive repair.

Characteristic of fog computing are as follows:

a) Contextual location awareness and low latency: The fog node sits as close to the IoT endpoint as possible to deliver distributed computing.
b) Geographic distribution: In sharp contrast to the more centralized cloud, the services and applications targeted by the fog nodes demand widely distributed deployments.
c) Deployment near IoT endpoints: Fog nodes are typically deployed in the presence of a large number of IoT endpoints. For example, typical metering deployments often see 3000 to 4000 nodes per gateway router, which also functions as the fog computing node.
d) Wireless communication between the fog and the IoT endpoint: Although it is possible to connect wired nodes, the advantages of fog are greatest when dealing with a large number of endpoints, and wireless access is the easiest way to achieve such scale.
e) Use for real-time interactions: Important fog applications involve real-time interactions rather than batch processing. Preprocessing of data in the fog nodes allows upper-layer applications to perform batch processing on a subset of the data.

Edge Computing

Fog computing solutions are being adopted by many industries, and efforts to develop distributed applications and analytics tools are being introduced at an accelerating pace. The natural place for a fog node is in the network device that sits closest to the IoT endpoints, and these nodes are typically spread throughout an IoT network.

IoT devices and sensors often have constrained resources, however, as compute capabilities increase. Some new classes of IoT endpoints have enough compute capabilities to perform at least low-level analytics and filtering to make basic decisions. For example, consider a water sensor on a fire hydrant. While a fog node sitting on an electrical pole in the distribution network may have an excellent view of all the fire hydrants in a local neighborhood, a node on each hydrant would have clear view of a water pressure drop on its own line and would be able to quickly generate an alert of a localized problem.

The fog node would have a wider view and would be able to ascertain whether the problem was more than just localized but was affecting the entire area. Another example is in the use of smart meters. Edge compute–capable meters are able to communicate with each other to share information on small subsets of the electrical distribution grid to monitor localized power quality and consumption, and they can inform a fog node of events that may pertain to only tiny sections of the grid. Models such as these help ensure the highest quality of power delivery to customers.

The Hierarchy of Edge, Fog, and Cloud

This model suggests a hierarchical organization of network, compute, and data storage resources. At each stage, data is collected, analyzed, and responded to when necessary, according to the capabilities of the resources at each layer. As data needs to be sent to the cloud, the latency becomes higher.

Edge and fog thus require an abstraction layer that allows applications to communicate with one another.

The abstraction layer :

- exposes a common set of APIs for monitoring, provisioning, and controlling the physical resources in a standardized way.
- requires a mechanism to support virtualization, with the ability to run multiple operating systems or service containers on physical devices to support multitenancy and application consistency across the IoT system.

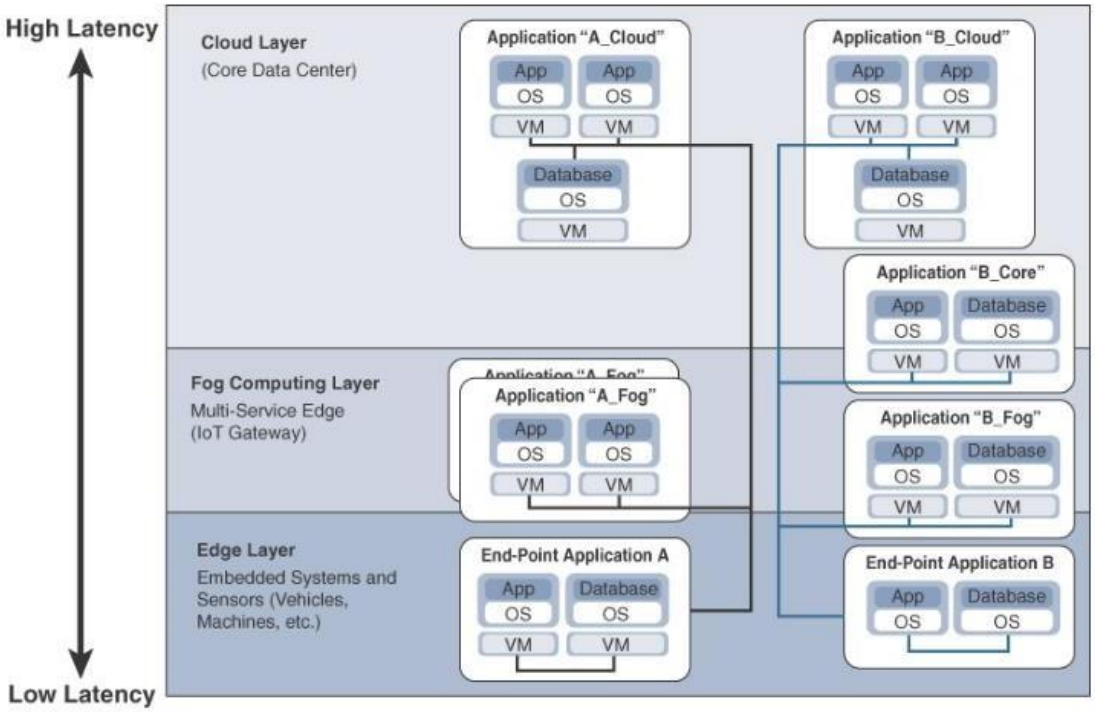


Figure 2-16 *Distributed Compute and Data Management Across an IoT System*

-

From an architectural standpoint, fog nodes closest to the network edge receive the data from IoT devices.

The fog IoT application then directs different types of data to the optimal place for analysis:

- The most time-sensitive data is analyzed on the edge or fog node closest to the things generating the data.
- Data that can wait seconds or minutes for action is passed along to an aggregation node for analysis and action.
- Data that is less time sensitive is sent to the cloud for historical analysis, big data analytics, and long-term storage. For example, each of thousands or hundreds of thousands of fog nodes might send periodic summaries of data to the cloud for historical analysis and storage.

MODULE 2
Smart Objects: The "Things" in IoT

Sensors are fundamental building blocks of IoT networks, they are the foundational elements found in smart objects—the "things" in the Internet of Things. Smart objects are any physical objects that contain embedded technology to sense and/or interact with their environment in a meaningful way by being interconnected and enabling communication among themselves or an external agent.

2.1 Sensors

A sensor does exactly as its name indicates: It senses. More specifically, a sensor measures some physical quantity and converts that measurement reading into a digital representation. That digital representation is typically passed to another device for transformation into useful data that can be consumed by intelligent devices or humans.

Naturally, a parallel can be drawn with humans and the use of their five senses to learn about their surroundings. Human senses do not operate independently in silos. Instead, they complement each other and compute together, empowering the human brain to make intelligent decisions. The brain is the ultimate decision maker.

There are different types of sensors available to measure virtually everything in the physical world and they are categorized as following:

- **Active or passive:** Sensors can be categorized based on whether they produce an energy output and typically require an external power supply (active) or whether they simply receive energy and typically require no external power supply (passive).
- **Invasive or non-invasive:** Sensors can be categorized based on whether a sensor is part of the environment it is measuring (invasive) or external to it (non-invasive).
- **Contact or no-contact**: Sensors can be categorized based on whether they require physical contact with what they are measuring (contact) or not (no-contact).
- **Absolute or relative:** Sensors can be categorized based on whether they measure on an absolute scale (absolute) or based on a difference with a fixed or variable reference value (relative).
- **Area of application:** Sensors can be categorized based on the specific industry or vertical where they are being used.
- **How sensors measure:** Sensors can be categorized based on the physical mechanism used to measure sensory input (for example, thermoelectric, electrochemical, piezoresistive, optic, electric, fluid mechanic, photoelastic).
- **What sensors measure:** Sensors can be categorized based on their applications or what physical variables they measure.

There are many other classification and taxonomic schemes for sensors, including those based on material, cost, design, and other factors. This type of categorization is shown in Table 3-1.

Table 3-1 Sensor Types

Sensor Types	Description	Examples
Position	A position sensor measures the position of an object; the position measurement can be either in absolute terms (absolute position sensor) or in relative terms (displacement sensor). Position sensors can be linear, angular, or multi-axis.	Potentiometer, inclinometer, proximity sensor
Occupancy and motion	Occupancy sensors detect the presence of people and animals in a surveillance area, while motion sensors detect movement of people and objects. The difference between the two is that occupancy sensors generate a signal even when a person is stationary, whereas motion sensors do not.	Electric eye, radar
Velocity and acceleration	Velocity (speed of motion) sensors may be linear or angular, indicating how fast an object moves along a straight line or how fast it rotates. Acceleration sensors measure changes in velocity.	Accelerometer, gyroscope
Force	Force sensors detect whether a physical force is applied and whether the magnitude of force is beyond a threshold.	Force gauge, viscometer, tactile sensor (touch sensor)
Pressure	Pressure sensors are related to force sensors, measuring force applied by liquids or gases. Pressure is measured in terms of force per unit area.	Barometer, Bourdon gauge, piezometer
Flow	Flow sensors detect the rate of fluid flow. They measure the volume (mass flow) or rate (flow velocity) of fluid that has passed through a system in a given period of time.	Anemometer, mass flow sensor, water meter
Acoustic	Acoustic sensors measure sound levels and convert that information into digital or analog data signals.	Microphone, geophone, hydrophone
Humidity	Humidity sensors detect humidity (amount of water vapor) in the air or a mass. Humidity levels can be measured in various ways: absolute humidity, relative humidity, mass ratio, and so on.	Hygrometer, humistor, soil moisture sensor
Biosensors	Biosensors detect various biological elements, such as organisms, tissues, cells, enzymes, antibodies, and nucleic acid.	Blood glucose biosensor, pulse oximetry, electrocardiograph

Light	Light sensors detect the presence of light (visible or invisible).	Infrared sensor, photodetector, flame detector
Radiation	Radiation sensors detect radiation in the environment. Radiation can be sensed by scintillating or ionization detection.	Geiger-Müller counter, scintillator, neutron detector
Temperature	Temperature sensors measure the amount of heat or cold that is present in a system. They can be broadly of two types: contact and non-contact. Contact temperature sensors need to be in physical contact with the object being sensed. Non-contact sensors do not need physical contact, as they measure temperature through convection and radiation.	Thermometer, calorimeter, temperature gauge
Chemical	Chemical sensors measure the concentration of chemicals in a system. When subjected to a mix of chemicals, chemical sensors are typically selective for a target type of chemical (for example, a CO_2 sensor senses only carbon dioxide).	Breathalyzer, olfactometer, smoke detector

Sensors come in all shapes and sizes and, as shown in Table 3-1, can measure all types of physical conditions. A fascinating use case to highlight the power of sensors and IoT is in the area of precision agriculture (smart farming), which uses a variety of technical advances to improve the efficiency, sustainability, and profitability of traditional farming practices. This includes the use of GPS and satellite aerial imagery for determining field viability; robots for high-precision planting, harvesting, irrigation, and so on; and real-time analytics and artificial intelligence to predict optimal crop yield, weather impacts, and soil quality.

The astounding volume of sensors is in large part due to their smaller size, their form factor, and their decreasing cost. These factors make possible the economic and technical feasibility of having an increased density of sensors in objects of all types. Perhaps the most significant accelerator for sensor deployments is mobile phones. More than a billion smart phones are sold each year, and each one has well over a dozen sensors inside it (see Figure 3-2), and that number continues to grow each year.

Figure 3-2 *Sensors in a Smart Phone*

It's fascinating to think that that a trillion-sensor economy is around the corner. Figure 3-3 shows the explosive year-over-year increase over the past several years and some bold predictions for sensor numbers in the upcoming years. There is a strong belief in the sensor industry that this number will eclipse a trillion in the next few years. In fact, many large players in the sensor industry have come together to form industry consortia, such as the TSensors Summits (www.tsensorssummit.org), to create a strategy and roadmap for a trillion-sensor economy. The trillion-sensor economy will be of such an unprecedented and unimaginable scale that it will change the world forever. This is the power of IoT.

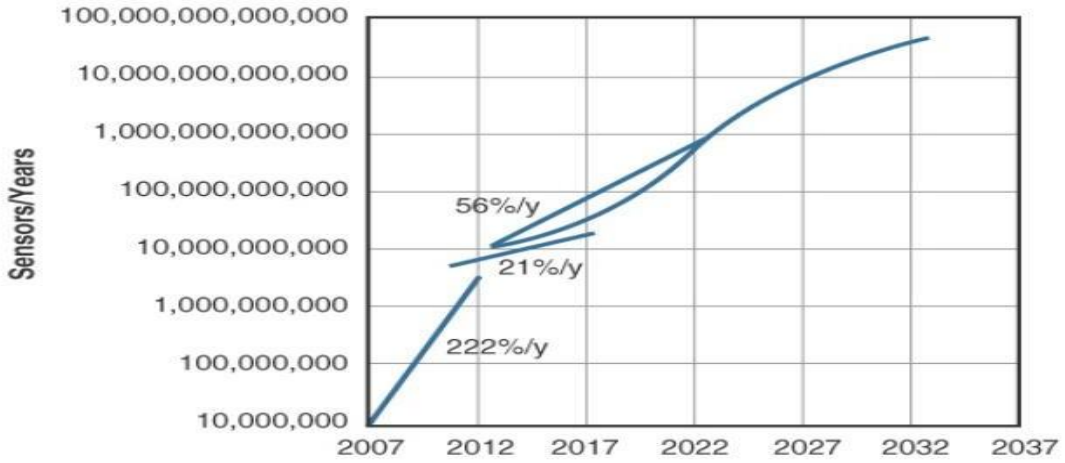


Figure 3-3 *Growth and Predictions in the Number of Sensors*

2.2 Actuators

Actuators are natural complements to sensors. Figure 3-4 demonstrates the symmetry and complementary nature of these two types of devices. Actuators receive some type of control signal (commonly an electric signal or digital command) that triggers a physical effect, usually some type of motion, force, and so on.

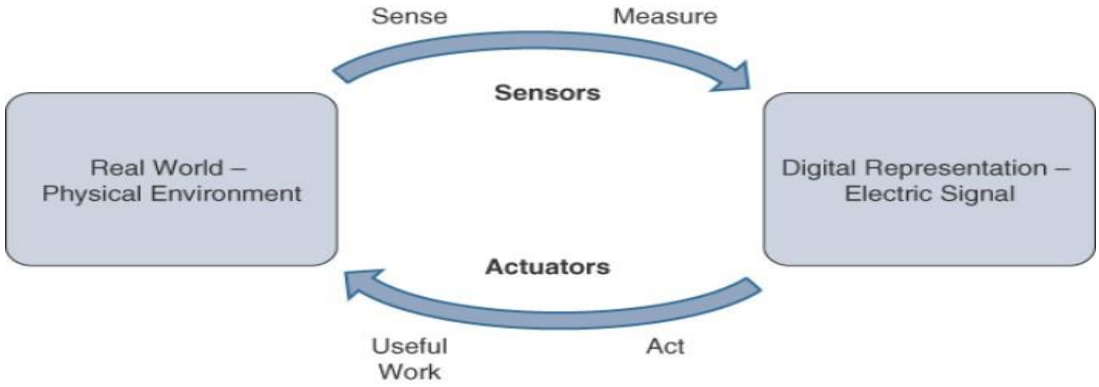


Figure 3-4 *How Sensors and Actuators Interact with the Physical World*

The previous section draws a parallel between sensors and the human senses. This parallel can be extended to include actuators, as shown in Figure 3-5. Humans use their five senses to sense and measure their environment. The sensory organs convert this sensory information into electrical impulses that the nervous system sends to the brain for processing. Likewise, IoT sensors are devices that sense and measure the physical world and (typically) signal their measurements as electric signals sent to some type of microprocessor or microcontroller for additional processing. The human brain signals motor function and movement, and the nervous system carries that information to the appropriate part of the muscular system. Correspondingly, a processor can send an electric signal to an actuator that translates the signal into some type of movement (linear, rotational, and so on) or useful work that changes or has a measurable impact on the physical world. This interaction between sensors, actuators, and processors and the similar functionality in biological systems is the basis for various technical fields, including robotics and biometrics.

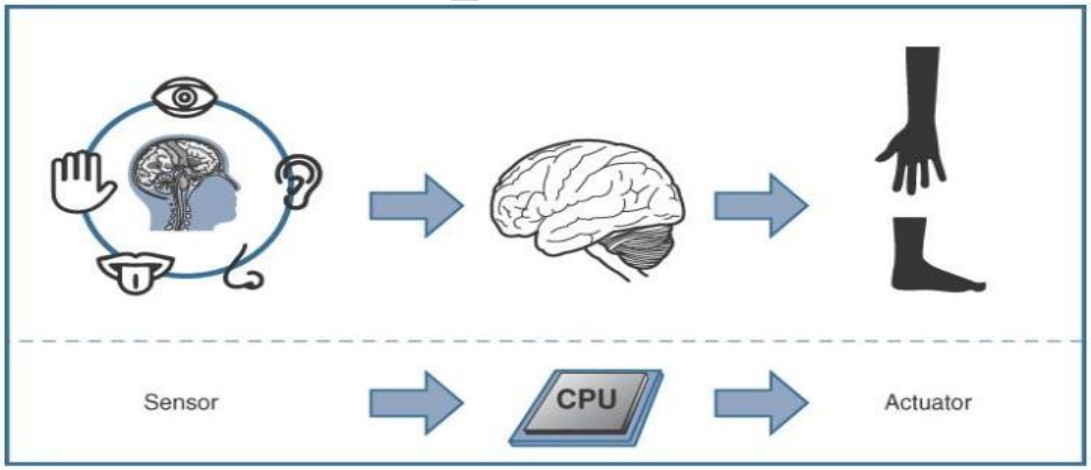


Figure 3-5 *Comparison of Sensor and Actuator Functionality with Humans*

Actuators vary greatly in function, size, design, and so on. Some common ways that they can be classified include the following:

- **Type of motion:** Actuators can be classified based on the type of motion they produce (for example, linear, rotary, one/two/three-axes).
- **Power:** Actuators can be classified based on their power output (for example, high power, low power, micro power)
- **Binary or continuous:** Actuators can be classified based on the number of stable-state outputs.
- **Area of application:** Actuators can be classified based on the specific industry or vertical where they are used.
- **Type of energy:** Actuators can be classified based on their energy type.

Categorizing actuators is quite complex, given their variety, so this is by no means an exhaustive list of classification schemes. The most commonly used classification is based on energy type. Table 3-2 shows actuators classified by energy type and some examples for each type.

Type	Examples
Mechanical actuators	Lever, screw jack, hand crank
Electrical actuators	Thyristor, biopolar transistor, diode
Electromechanical actuators	AC motor, DC motor, step motor
Electromagnetic actuators	Electromagnet, linear solenoid
Hydraulic and pneumatic actuators	Hydraulic cylinder, pneumatic cylinder, piston, pressure control valves, air motors
Smart material actuators (includes thermal and magnetic actuators)	Shape memory alloy (SMA), ion exchange fluid, magnetorestrictive material, bimetallic strip, piezoelectric bimorph
Micro- and nanoactuators	Electrostatic motor, microvalve, comb drive

Table 3-2 *Actuator Classification by Energy Type*

Whereas sensors provide the information, actuators provide the action. The most interesting use cases for IoT are those where sensors and actuators work together in an intelligent, strategic, and complementary fashion. This powerful combination can be used to solve everyday problems by simply elevating the data that sensors provide to actionable insight that can be acted on by work- producing actuators.

The precision agriculture example can demonstrate how actuators can be complement and enhance a sensor-only solution. For example, the smart sensors used to evaluate soil quality (by measuring a variety of soil, temperature, and plant characteristics) can be connected with electrically or pneumatically controlled valve actuators that control water, pesticides, fertilizers, herbicides, and so on. Intelligently triggering a high-precision actuator based on well-defined sensor readings of temperature, pH, soil/air humidity, nutrient levels, and so on to deliver a highly optimized and custom environment-specific solution is truly smart farming.

3.3 Smart Objects

Smart objects are, quite simply, the building blocks of IoT. They are what transform everyday objects into a network of intelligent objects that are able to learn from and interact with their environment in a meaningful way. It can't be stressed enough that the real power of smart objects in IoT comes from being networked together rather than being isolated as standalone objects. This ability to communicate over a network has a multiplicative effect and allows for very sophisticated correlation and interaction between disparate smart objects. For instance, recall the smart farming sensors described previously. If a sensor is a standalone device that simply measures the humidity of the soil, it is interesting and useful, but it isn't revolutionary. If that same sensor is connected as part of an intelligent network that is able to coordinate intelligently with actuators to trigger irrigation systems as needed based on those sensor readings. Extending that even further, imagine that the coordinated sensor/actuator set is intelligently interconnected with other sensor/actuator sets to further coordinate fertilization, pest control, and so on—and even communicate with an intelligent backend to calculate crop yield potential. This now starts to look like a complete system that begins to unlock the power of IoT and provides the intelligent automation to expect from such a revolutionary technology.

3.3.1 Smart Objects: A Definition

The term smart object, despite some semantic differences, is often used interchangeably with terms such as smart sensor, smart device, IoT device, intelligent device, thing, smart thing, intelligent node, intelligent thing and intelligent product. A smart object, is described as a device that has, at a minimum, one of the following four defining characteristics (see Figure 3-7):

- **Processing unit:** A smart object has some type of processing unit for acquiring data, processing and analyzing sensing information received by the sensor(s), coordinating control signals to any actuators, and controlling a variety of functions on the smart object, including the communication and power systems. The most common is a microcontroller because of its small form factor, flexibility, programming simplicity, ubiquity, low power consumption, and low cost.
- **Sensor(s) and/or actuator(s):** A smart object is capable of interacting with the physical world through sensors and actuators. A smart object does not need to contain both sensors and actuators. In fact, a smart object can contain one or multiple sensors and/or actuators, depending upon the application.
- **Communication device:** The communication unit is responsible for connecting a smart object with other smart objects and the outside world (via the network). Communication devices for smart objects can be either wired or wireless. Overwhelmingly, in IoT networks smart objects are wirelessly interconnected for a number of reasons, including cost, limited infrastructure availability, and ease of deployment. There are myriad different communication protocols for smart objects.

- **Power source:** Smart objects have components that need to be powered. Interestingly, the most significant power consumption usually comes from the communication unit of a smart object. Typically, smart objects are limited in power, are deployed for a very long time, and are not easily accessible. This combination, especially when the smart object relies on battery power, implies that power efficiency, judicious power management, sleep modes, ultra-low power consumption hardware, and so on are critical design elements. For long-term deployments where smart objects are, for all practical purposes, inaccessible, power is commonly obtained from scavenger sources (solar, piezoelectric, and so on) or is obtained in a hybridized manner, also tapping into infrastructure power.

3.3.2 Trends in Smart Objects

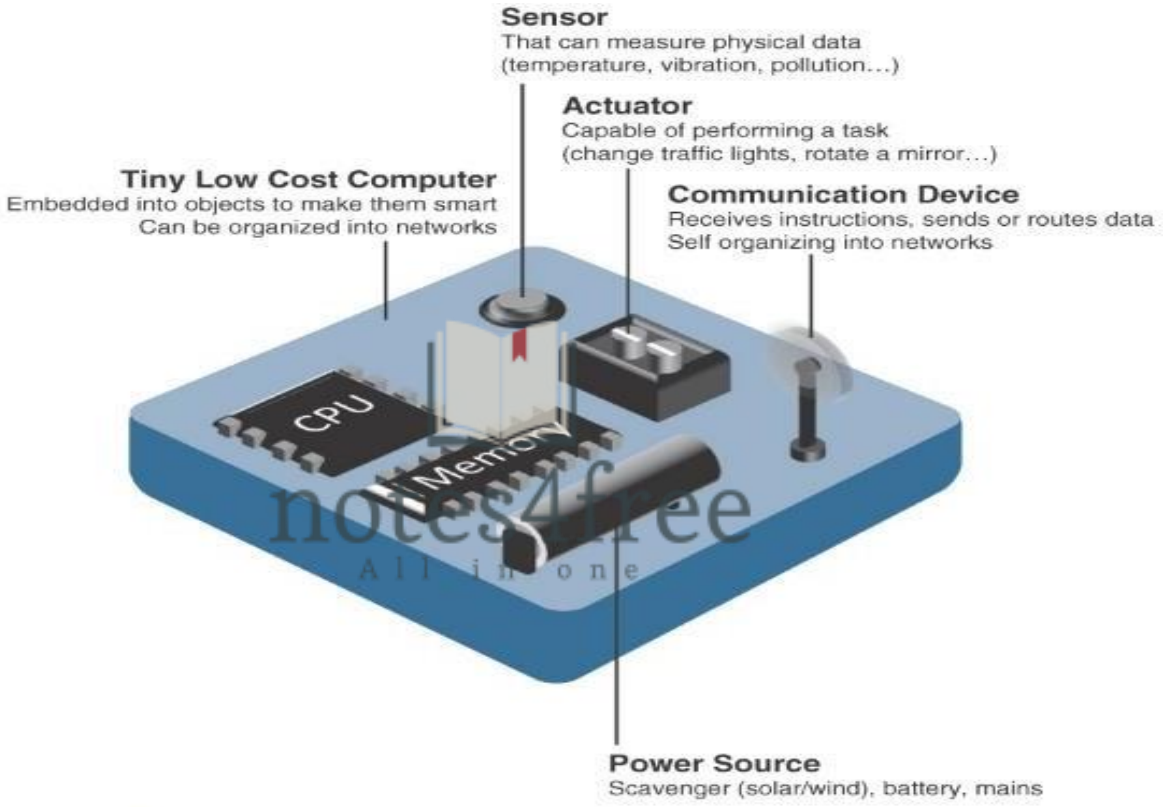


Figure 3-7 *Characteristics of a Smart Object*

Smart objects vary wildly in function, technical requirements, form factor, deployment conditions, and so on. There are certain important macro trends that can be infered from recent and planned future smart object deployments. Of course, these do not apply to all smart objects because there will always be application-dependent variability, but these are broad generalizations and trends impacting IoT:

- **Size is decreasing:** As discussed earlier, in reference to MEMS, there is a clear trend of ever-decreasing size. Some smart objects are so small they are not even visible to the naked eye. This reduced size makes smart objects easier to embed in everyday objects.
- **Power consumption is decreasing:** The different hardware components of a smart object continually consume less power. This is especially true for sensors, many of which are

completely passive. Some battery-powered sensors last 10 or more years without battery replacement.

- **Processing power is increasing:** Processors are continually getting more powerful and smaller. This is a key advancement for smart objects, as they become increasingly complex and connected.
- **Communication capabilities are improving:** It's no big surprise that wireless speeds are continually increasing, but they are also increasing in range. IoT is driving the development of more and more specialized communication protocols covering a greater diversity of use cases and environments.
- **Communication is being increasingly standardized:** There is a strong push in the industry to develop open standards for IoT communication protocols. In addition, there are more and more open source efforts to advance IoT.

These trends in smart objects begin to paint a picture of increasingly sophisticated devices that are able to perform increasingly complex tasks with greater efficiency. The power of IoT is truly unlocked when smart objects are networked together in sensor/actuator networks.

3.4 Sensor Networks

A sensor/actuator network (SANET), as the name suggests, is a network of sensors that sense and measure their environment and/or actuators that act on their environment. The sensors and/or actuators in a SANET are capable of communicating and cooperating in a productive manner. Effective and well-coordinated communication and cooperation is a prominent challenge, primarily because the sensors and actuators in SANETs are diverse, heterogeneous, and resource-constrained.

SANETs offer highly coordinated sensing and actuation capabilities. Smart homes are a type of SANET that display this coordination between distributed sensors and actuators. For example, smart homes can have temperature sensors that are strategically networked with heating, ventilation, and air-conditioning (HVAC) actuators. When a sensor detects a specified temperature, this can trigger an actuator to take action and heat or cool the home as needed.

While such networks can theoretically be connected in a wired or wireless fashion, the fact that SANETs are typically found in the "real world" means that they need an extreme level of deployment flexibility. For example, smart home temperature sensors need to be expertly located in strategic locations throughout the home, including at HVAC entry and exit points.

The following are some advantages and disadvantages that a wireless-based solution offers:

Advantages:

- Greater deployment flexibility (especially in extreme environments or hard-to-reach places)
- Simpler scaling to a large number of nodes
- Lower implementation costs
- Easier long-term maintenance
- Effortless introduction of new sensor/actuator nodes
- Better equipped to handle dynamic/rapid topology changes

Disadvantages:

- Potentially less secure (for example, hijacked access points)
- Typically lower transmission speeds
- Greater level of impact/influence by environment

3.4.2 Wireless Sensor Networks (WSNs)

Wireless sensor networks are made up of wirelessly connected smart objects, which are sometimes referred to as motes. The fact that there is no infrastructure to consider with WSNs is surely a powerful advantage for flexible deployments, but there are a variety of design constraints to consider with these wirelessly connected smart objects. Figure 3-8 illustrates some of these assumptions and constraints usually involved in WSNs.

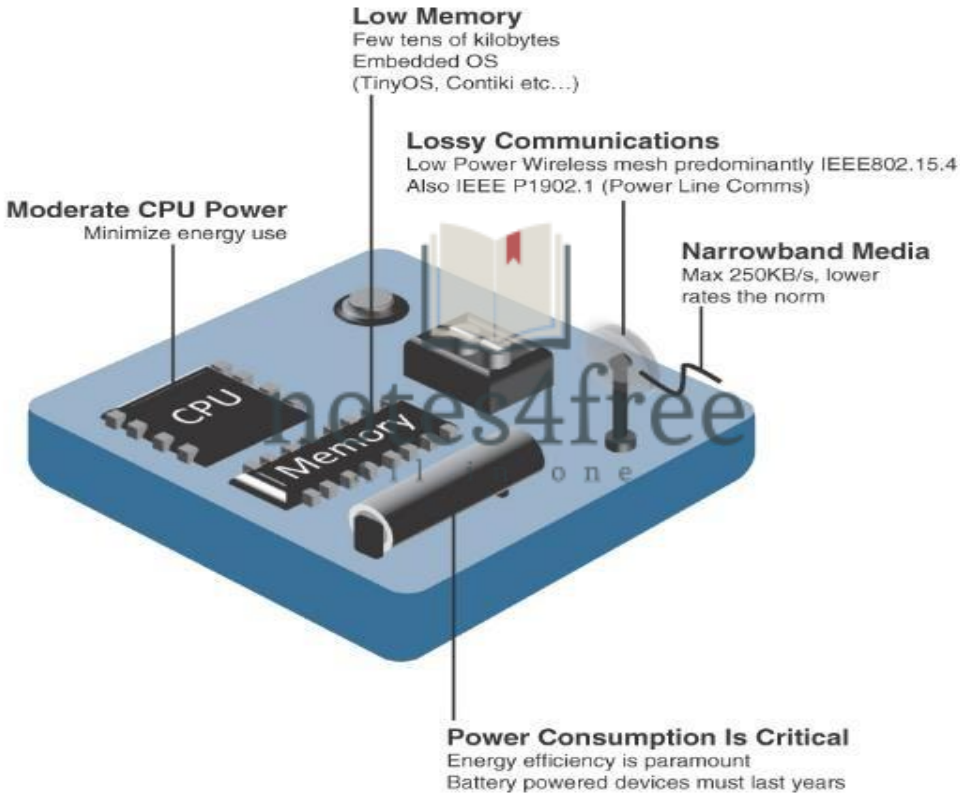


Figure 3-8 *Design Constraints for Wireless Smart Objects*

The following are some of the most significant limitations of the smart objects in WSNs:

- Limited processing power
- Limited memory
- Lossy communication
- Limited transmission speeds
- Limited power

These limitations greatly influence how WSNs are designed, deployed, and utilized. The fact that individual sensor nodes are typically so limited is a reason that they are often deployed in very large numbers. As the cost of sensor nodes continues to decline, the ability to deploy highly redundant sensors becomes increasingly feasible. Because many sensors are very inexpensive and correspondingly inaccurate, the ability to deploy smart objects redundantly allows for increased accuracy.

Such large numbers of sensors permit the introduction of hierarchies of smart objects. Such a hierarchy provides, among other organizational advantages, the ability to aggregate similar sensor readings from sensor nodes that are in close proximity to each other. Figure 3-9 shows an example of such a data aggregation function in a WSN where temperature readings from a logical grouping of temperature sensors are aggregated as an average temperature reading.

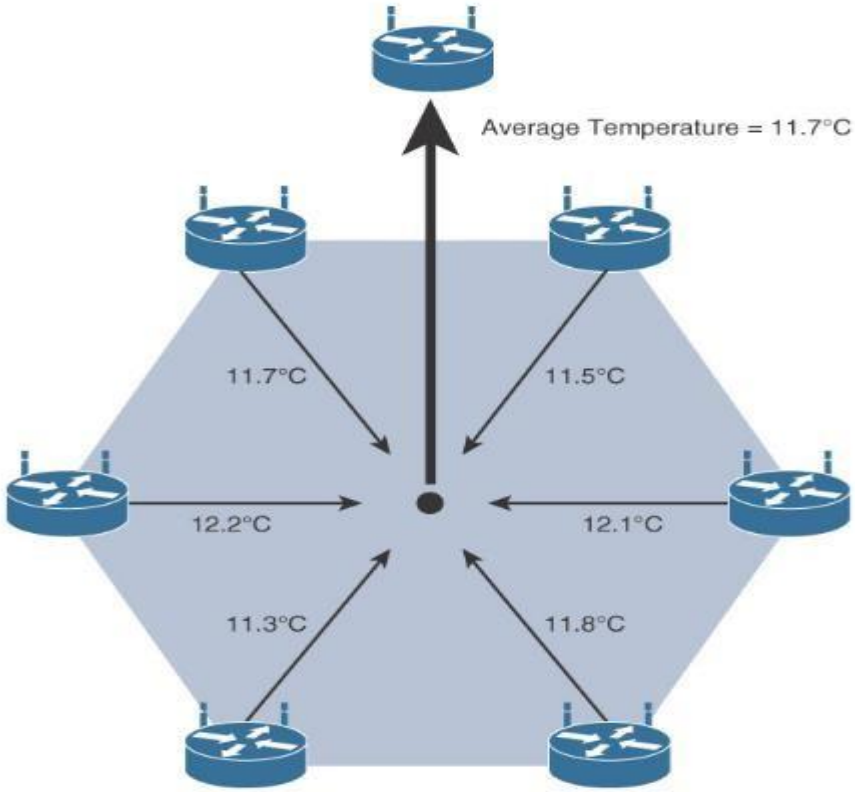


Figure 3-9 *Data Aggregation in Wireless Sensor Networks*

These data aggregation techniques are helpful in reducing the amount of overall traffic (and energy) in WSNs with very large numbers of deployed smart objects. This data aggregation at the network edges is where fog and mist computing, While there are certain instances in which sensors continuously stream their measurement data, this is typically not the case. Wirelessly connected smart objects generally have one of the following two communication patterns:

- **Event-driven:** Transmission of sensory information is triggered only when a smart object detects a particular event or predetermined threshold.
- **Periodic:** Transmission of sensory information occurs only at periodic intervals.

The decision of which of these communication schemes is used depends greatly on the specific application. For example, in some medical use cases, sensors periodically send postoperative vitals, such as temperature or blood pressure readings. In other medical use cases, the same blood pressure

or temperature readings are triggered to be sent only when certain critically low or high readings are measured.

3.4.3 Communication Protocols for Wireless Sensor Networks

WSNs are becoming increasingly heterogeneous, with more sophisticated interactions. This heterogeneity is shown in a variety of ways. WSNs are also evolving from single-purpose networks to more flexible multipurpose networks that can use specific sensor types for multiple different applications at any given time. Imagine a WSN that has multiple types of sensors, and one of those types is a temperature sensor that can be flexibly used concurrently for environmental applications, weather applications, and smart farming applications.

Coordinated communication with sophisticated interactions by constrained devices within such a heterogeneous environment is quite a challenge. The protocols governing the communication for WSNs must deal with the inherent defining characteristics of WSNs and the constrained devices within them. Any communication protocol must be able to scale to a large number of nodes. Likewise, when selecting a communication protocol, care must be taken to account the requirements of the specific application and consider any trade-offs the communication protocol offers between power consumption, maximum transmission speed, range, tolerance for packet loss, topology optimization, security, and so on. The fact that WSNs are often deployed outdoors in harsh and unpredictable environments adds yet another variable to consider because obviously not all communication protocols are designed to be equally rugged.

Wireless sensor networks interact with their environment. Sensors often produce large amounts of sensing and measurement data that needs to be processed. This data can be processed locally by the nodes of a WSN or across zero or more hierarchical levels in IoT networks. Communication protocols need to facilitate routing and message handling for this data flow between sensor nodes as well as from sensor nodes to optional gateways, edge compute, or centralized cloud compute. IoT communication protocols for WSNs thus straddle the entire protocol stack. Ultimately, they are used to provide a platform for a variety of IoT smart services.

3.5 Connecting Smart Objects

IoT devices and sensors must be connected to the network for their data to be utilized. In addition to the wide range of sensors, actuators, and smart objects that make up IoT, there are also a number of different protocols used to connect them. Here the characteristics and communications criteria that are important for the technologies that smart objects employ for their connectivity, along with a deeper dive into some of the major technologies being deployed today.

3.5.1 Communications Criteria

In the world of connecting “things,” a large number of wired and wireless access technologies are available or under development. Before reviewing some of these access technologies, it is important to talk about the criteria to use in evaluating them for various use cases and system solutions.

Wireless communication is prevalent in the world of smart object connectivity, mainly because it eases deployment and allows smart objects to be mobile, changing location without losing connectivity. The following sections take this into account as they discuss various criteria. In addition, wired connectivity considerations are mentioned when applicable.

3.5.1.1 Range

How far does the signal need to be propagated? That is, what will be the area of coverage for a selected wireless technology? Should indoor versus outdoor deployments be differentiated? Very often, these are the questions asked when discussing wired and wireless access technologies. The simplest approach to answering these types of questions is to categorize these technologies as shown in Figure 4-1, breaking them down into the following ranges:

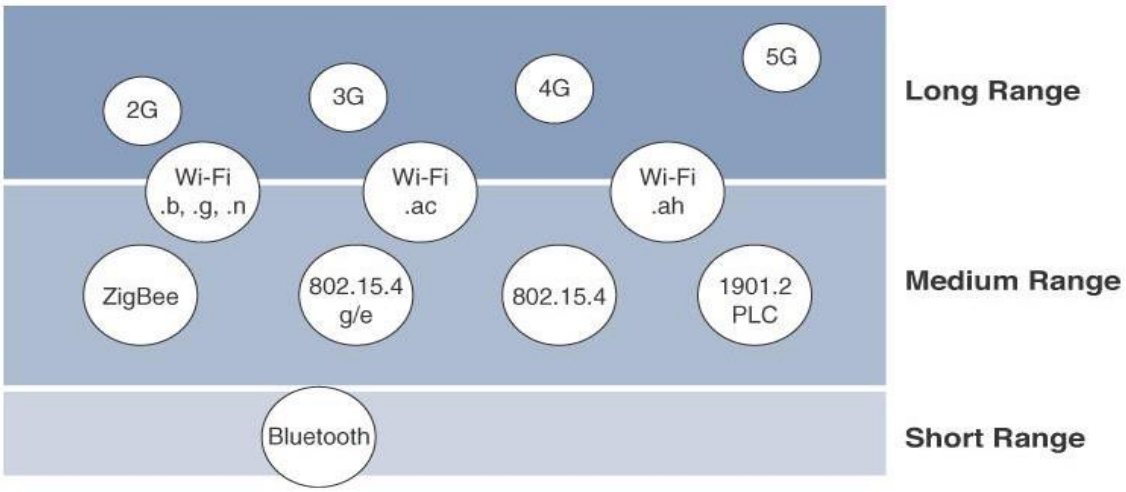


Figure 4-1 *Wireless Access Landscape*

- **Short range:** The classical wired example is a serial cable. Wireless short-range technologies are often considered as an alternative to a serial cable, supporting tens of meters of maximum distance between two devices. Examples of short-range wireless technologies are IEEE 802.15.1 Bluetooth and IEEE 802.15.7 Visible Light Communications (VLC). These short-range communication methods are found in only a minority of IoT installations. In some cases, they are not mature enough for production deployment.
- **Medium range:** This range is the main category of IoT access technologies. In the range of tens to hundreds of meters, many specifications and implementations are available. The maximum distance is generally less than 1 mile between two devices, although RF technologies do not have real maximum distances defined, as long as the radio signal is transmitted and received in the scope of the applicable specification. Examples of medium- range wireless technologies include IEEE 802.11 Wi-Fi, IEEE 802.15.4, and 802.15.4g WPAN. Wired technologies such as IEEE 802.3 Ethernet and IEEE 1901.2 Narrowband Power Line Communications (PLC) may also be classified as medium range, depending on their physical media characteristics.

- **Long range:** Distances greater than 1 mile between two devices require long-range technologies. Wireless examples are cellular (2G, 3G, 4G) and some applications of outdoor IEEE 802.11 Wi-Fi and Low-Power Wide-Area (LPWA) technologies. LPWA communications have the ability to communicate over a large area without consuming much power. These technologies are therefore ideal for battery-powered IoT sensors. Found mainly in industrial networks, IEEE 802.3 over optical fiber and IEEE 1901 Broadband Power Line Communications are classified as long range but are not really considered IoT access technologies.

3.5.1.2 Frequency Bands

Radio spectrum is regulated by countries and/or organizations, such as the International Telecommunication Union (ITU) and the Federal Communications Commission (FCC). These groups define the regulations and transmission requirements for various frequency bands. For example, portions of the spectrum are allocated to types of telecommunications such as radio, television, military, and so on.

Focusing on IoT access technologies, the frequency bands leveraged by wireless communications are split between licensed and unlicensed bands. Licensed spectrum is generally applicable to IoT long-range access technologies and allocated to communications infrastructures deployed by services providers, public services (for example, first responders, military), broadcasters, and utilities.

An important consideration for IoT access infrastructures that wish to utilize licensed spectrum is that users must subscribe to services when connecting their IoT devices. This adds more complexity to a deployment involving large numbers of sensors and other IoT devices, but in exchange for the subscription fee, the network operator can guarantee the exclusivity of the frequency usage over the target area and can therefore sell a better guarantee of service.

The ITU has also defined unlicensed spectrum for the industrial, scientific, and medical (ISM) portions of the radio bands. These frequencies are used in many communications technologies for short-range devices (SRDs). Unlicensed means that no guarantees or protections are offered in the ISM bands for device communications. For IoT access, these are the most well-known ISM bands:

- 2.4 GHz band as used by IEEE 802.11b/g/n Wi-Fi
- IEEE 802.15.1 Bluetooth
- IEEE 802.15.4 WPAN

An unlicensed band, such as those in the ISM range of frequencies, is not unregulated. National and regional regulations exist for each of the allocated frequency bands (much as with the licensed bands). These regulations mandate device compliance on parameters such as transmit power, duty cycle and dwell time, channel bandwidth, and channel hopping.

Unlicensed spectrum is usually simpler to deploy than licensed because it does not require a service provider. However, it can suffer from more interference because other devices may be competing for the same frequency in a specific area. This becomes a key element in decisions for IoT deployments.

Should an IoT infrastructure utilize unlicensed spectrum available for private networks or licensed frequencies that are dependent on a service provider? Various LPWA technologies are taking on a greater importance when it comes to answering this question. In addition to meeting low power requirements, LPWA communications are able to cover long distances that in the past required the licensed bands offered by service providers for cellular devices.

3.5.1.3 Power Consumption

While the definition of IoT device is very broad, there is a clear delineation between powered nodes and battery-powered nodes. A powered node has a direct connection to a power source, and communications are usually not limited by power consumption criteria. However, ease of deployment of powered nodes is limited by the availability of a power source, which makes mobility more complex.

Battery-powered nodes bring much more flexibility to IoT devices. These nodes are often classified by the required lifetimes of their batteries. Does a node need 10 to 15 years of battery life, such as on water or gas meters? Or is a 5- to 7-year battery life sufficient for devices such as smart parking sensors? Their batteries can be changed or the devices replaced when a street gets resurfaced. For devices under regular maintenance, a battery life of 2 to 3 years is an option.

IoT wireless access technologies must address the needs of low power consumption and connectivity for battery-powered nodes. This has led to the evolution of a new wireless environment known as Low-Power Wide-Area (LPWA). Obviously, it is possible to run just about any wireless technology on batteries. However, in reality, no operational deployment will be acceptable if hundreds of batteries must be changed every month.

Wired IoT access technologies consisting of powered nodes are not exempt from power optimization. In the case of deployment of smart meters over PLC, the radio interface on meters can't consume 5 to 10 watts of power, or this will add up to a 20-million-meter deployment consuming 100 to 200 megawatts of energy for communications.

3.5.1.4 Topology

Among the access technologies available for connecting IoT devices, three main topology schemes are dominant: star, mesh, and peer-to-peer. For long-range and short-range technologies, a star topology is prevalent, as seen with cellular, LPWA, and Bluetooth networks. Star topologies utilize a single central base station or controller to allow communications with endpoints.

For medium-range technologies, a star, peer-to-peer, or mesh topology is common, as shown in Figure 4-2. Peer-to-peer topologies allow any device to communicate with any other device as long as they are in range of each other. Obviously, peer-to-peer topologies rely on multiple full-function devices. Peer-to-peer topologies enable more complex formations, such as a mesh networking topology.

A mesh topology helps cope with low transmit power, searching to reach a greater overall distance, and coverage by having intermediate nodes relaying traffic for other nodes. Mesh topology requires the implementation of a Layer 2 forwarding protocol known as mesh-under or a Layer 3 forwarding protocol referred to as mesh-over on each intermediate node. An intermediate node or full-function device (FFD) is simply a node that interconnects other nodes. A node that doesn't interconnect or relay the traffic of other nodes is known as a leaf node, or reduced-function device (RFD).

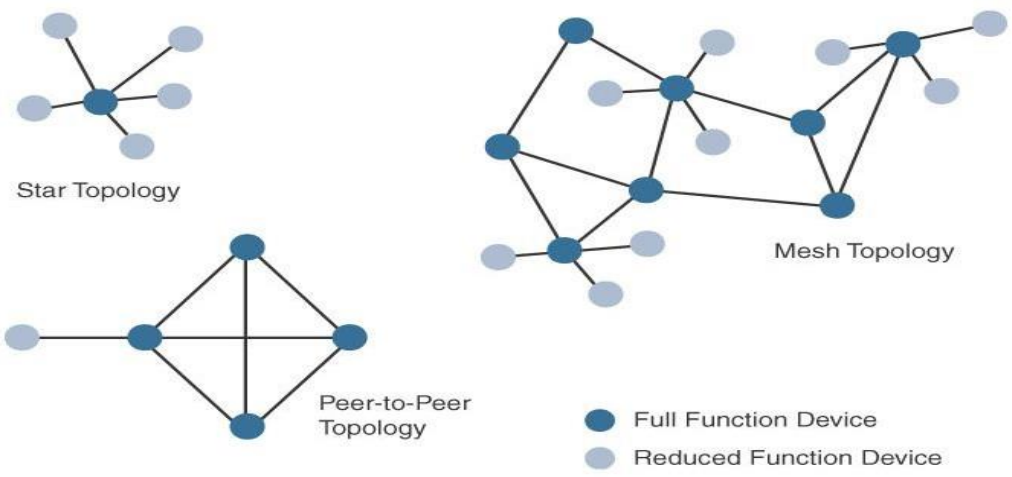


Figure 4-2 *Star, Peer-to-Peer, and Mesh Topologies*

3.5 Constrained Devices

The Internet Engineering Task Force (IETF) acknowledges in RFC 7228 that different categories of IoT devices are deployed. While categorizing the class of IoT nodes is a perilous exercise, with computing, memory, storage, power, and networking continuously evolving and improving, RFC 7228 gives some definitions of constrained nodes. These definitions help differentiate constrained nodes from unconstrained nodes, such as servers, desktop or laptop computers, and powerful mobile devices such as smart phones.According to RFC 7228, constrained nodes can be broken down into the classes defined in Table 4-1.

Class	Definition
Class 0	This class of nodes is severely constrained, with less than 10 KB of memory and less than 100 KB of Flash processing and storage capability. These nodes are typically battery powered. They do not have the resources required to directly implement an IP stack and associated security mechanisms. An example of a Class 0 node is a push button that sends 1 byte of information when changing its status. This class is particularly well suited to leveraging new unlicensed LPWA wireless technology.
Class 1	While greater than Class 0, the processing and code space characteristics (approximately 10 KB RAM and approximately 100 KB Flash) of Class 1 are still lower than expected for a complete IP stack implementation. They cannot easily communicate with nodes employing a full IP stack. However, these nodes can implement an optimized stack specifically designed for constrained nodes, such as Constrained Application Protocol (CoAP). This allows Class 1 nodes to engage in meaningful conversations with the network without the help of a gateway, and provides support for the necessary security functions. Environmental sensors are an example of Class 1 nodes.
Class 2	Class 2 nodes are characterized by running full implementations of an IP stack on embedded devices. They contain more than 50 KB of memory and 250 KB of Flash, so they can be fully integrated in IP networks. A smart power meter is an example of a Class 2 node.

Table 4-1 *Classes of Constrained Nodes, as Defined by RFC 7228*

3.6.1 Constrained-Node Networks

While several of the IoT access technologies, such as Wi-Fi and cellular, are applicable to laptops, smart phones, and some IoT devices, some IoT access technologies are more suited to specifically connect constrained nodes. Typical examples are IEEE 802.15.4 and 802.15.4g RF, IEEE 1901.2a PLC, LPWA, and IEEE 802.11ah access technologies.

Constrained-node networks are often referred to as **Low-power and Lossy Networks (LLNs).** Low-power in the context of LLNs refers to the fact that nodes must cope with the requirements from powered and battery-powered constrained nodes. Lossy networks indicates that network performance may suffer from interference and variability due to harsh radio environments. Layer 1 and Layer 2 protocols that can be used for constrained-node networks must be evaluated in the context of the following characteristics for use-case applicability: data rate and throughput, latency and determinism, and overhead and payload.

1. **Data Rate and Throughput**

The data rates available from IoT access technologies range from 100 bps with protocols such as Sigfox to tens of megabits per second with technologies such as LTE and IEEE 802.11ac. However, the actual throughput is less—sometimes much less—than the data rate. Therefore, understanding the bandwidth requirements of a particular technology, its applicability to given use cases, the capacity

planning rules, and the expected real throughput are important for proper network design and successful production deployment.

Technologies not particularly designed for IoT, such as cellular and Wi-Fi, match up well to IoT applications with high bandwidth requirements. For example, nodes involved with video analytics have a need for high data rates. These nodes are found in retail, airport, and smart cities environments for detecting events and driving actions. Because these types of IoT endpoints are not constrained in terms of computing or network bandwidth, the design guidelines tend to focus on application requirements, such as latency and determinism.

Short-range technologies can also provide medium to high data rates that have enough throughput to connect a few endpoints. For example, Bluetooth sensors that are now appearing on connected wearables fall into this category. The IoT access technologies developed for constrained nodes are optimized for low power consumption, but they are also limited in terms of data rate, which depends on the selected frequency band, and throughput.

A discussion of data rate and bandwidth in LLNs must include a look at real throughput, or "goodput," as seen by the application. While it may not be important for constrained nodes that send only one message a day, real throughput is often very important for constrained devices implementing an IP stack. In this case, throughput is a lower percentage of the data rate, even if the node gets the full constrained network at a given time.

Another characteristic of IoT devices is that a majority of them initiate the communication. Upstream traffic toward an application server is usually more common than downstream traffic from the application server. Understanding this behavior also helps when deploying an IoT access technology, such as cellular, that is asymmetrical because the upstream bandwidth must be considered a key parameter for profiling the network capacity.

Latency and Determinism

Much like throughput requirements, latency expectations of IoT applications should be known when selecting an access technology. This is particularly true for wireless networks, where packet loss and retransmissions due to interference, collisions, and noise are normal behaviors.

On constrained networks, latency may range from a few milliseconds to seconds, and applications and protocol stacks must cope with these wide-ranging values. For example, UDP at the transport layer is strongly recommended for IP endpoints communicating over LLNs. In the case of mesh topologies, if communications are needed between two devices inside the mesh, the forwarding path may call for some routing optimization, which is available using the IPv6 RPL protocol.

Overhead and Payload

When considering constrained access network technologies, it is important to review the MAC payload size characteristics required by applications. The minimum IPv6 MTU size is expected to be

1280 bytes. Therefore, the fragmentation of the IPv6 payload has to be taken into account by link layer access protocols with smaller MTUs.

For example, the payload size for IEEE 802.15.4 is 127 bytes and requires an IPv6 payload with a minimum MTU of 1280 bytes to be fragmented. On the other hand, IEEE 802.15.4g enables payloads up to 2048 bytes, easing the support of the IPv6 minimum MTU of 1280 bytes.

Most LPWA technologies offer small payload sizes. These small payload sizes are defined to cope with the low data rate and time over the air or duty cycle requirements of IoT nodes and sensors. For example, payloads may be as little as 19 bytes using LoRaWAN technology or up to 250 bytes, depending on the adaptive data rate (ADR). While this doesn't preclude the use of an IPv6/6LoWPAN payload, as seen on some endpoint implementations, these types of protocols are better suited to Class 0 and 1 nodes, as defined in RFC 7228.

3.6 IoT Access Technologies

The technologies highlighted here are the ones that are seen as having market and/or mind share. Therefore, it is necessary to have a basic familiarity with them as they are fundamental to many IoT conversations. For each of the IoT access technologies discussed here, a common information set is being provided. Particularly, the following topics are addressed for each IoT access technology:

- **Standardization and alliances:** The standards bodies that maintain the protocols for a technology
- **Physical layer:** The wired or wireless methods and relevant frequencies
- **MAC layer:** Considerations at the Media Access Control (MAC) layer, which bridges the physical layer with data link control
- **Topology:** The topologies supported by the technology
- **Security:** Security aspects of the technology
- **Competitive technologies:** Other technologies that are similar and may be suitable alternatives to the given technology

3.6.1 IEEE 802.15.4

IEEE 802.15.4 is a wireless access technology for low-cost and low-data-rate devices that are powered or run on batteries. In addition to being low cost and offering a reasonable battery life, this access technology enables easy installation using a compact protocol stack while remaining both simple and flexible. Several network communication stacks, including deterministic ones, and profiles leverage this technology to address a wide range of IoT use cases in both the consumer and business markets. IEEE 802.15.4 is commonly found in the following types of deployments:

- Home and building automation
- Automotive networks
- Industrial wireless sensor networks
- Interactive toys and remote controls

Criticisms of IEEE 802.15.4 often focus on its MAC reliability, unbounded latency, and susceptibility to interference and multipath fading. The negatives around reliability and latency often have to do with the Collision Sense Multiple Access/Collision Avoidance (CSMA/CA) algorithm. CSMA/CA is an access method in which a device "listens" to make sure no other devices are transmitting before starting its own transmission.

Standardization and Alliances

IEEE 802.15.4 or IEEE 802.15 Task Group 4 defines low-data-rate PHY and MAC layer specifications for wireless personal area networks (WPAN). This standard has evolved over the years and is a well-known solution for low-complexity wireless devices with low data rates that need many months or even years of battery life.

While there is no alliance or promotion body for IEEE 802.15.4 per se, the IEEE 802.15.4 PHY and MAC layers are the foundations for several networking protocol stacks. These protocol stacks make use of 802.15.4 at the physical and link layer levels, but the upper layers are different. These protocol stacks are promoted separately through various organizations and often commercialized. Some of the most well-known protocol stacks based on 802.15.4 are highlighted in Table 4-2.

Protocol	Description
ZigBee	Promoted through the ZigBee Alliance, ZigBee defines upper-layer components (network through application) as well as application profiles. Common profiles include building automation, home automation, and healthcare. ZigBee also defines device object functions, such as device role, device discovery, network join, and security. For more information on ZigBee, see the ZigBee Alliance webpage, at www.zigbee.org. ZigBee is also discussed in more detail later in the next Section.
6LoWPAN	6LoWPAN is an IPv6 adaptation layer defined by the IETF 6LoWPAN working group that describes how to transport IPv6 packets over IEEE 802.15.4 layers. RFCs document header compression and IPv6 enhancements to cope with the specific details of IEEE 802.15.4. (For more information on 6LoWPAN, see Chapter 5.)

ZigBee IP	An evolution of the ZigBee protocol stack, ZigBee IP adopts the 6LoWPAN adaptation layer, IPv6 network layer, and RPL routing protocol. In addition, it offers improvements to IP security. ZigBee IP is discussed in more detail later in this chapter.
ISA100.11a	ISA100.11a is developed by the International Society of Automation (ISA) as "Wireless Systems for Industrial Automation: Process Control and Related Applications." It is based on IEEE 802.15.4-2006, and specifications were published in 2010 and then as IEC 62734. The network and transport layers are based on IETF 6LoWPAN, IPv6, and UDP standards.
WirelessHART	WirelessHART, promoted by the HART Communication Foundation, is a protocol stack that offers a time-synchronized, self-organizing, and self-healing mesh architecture, leveraging IEEE 802.15.4-2006 over the 2.4 GHz frequency band. A good white paper on WirelessHART can be found at http://www.emerson.com/resource/blob/system-engineering-guidelines-iec-62591-wirelesshart--data-79900.pdf
Thread	Constructed on top of IETF 6LoWPAN/IPv6, Thread is a protocol stack for a secure and reliable mesh network to connect and control products in the home. Specifications are defined and published by the Thread Group at www.threadgroup.org.

Table 4-2 *Protocol Stacks Utilizing IEEE 802.15.4*

Because of its relatively long history compared to the others, ZigBee is one of the most well-known protocols listed in Table 4-2. In addition, ZigBee has continued to evolve over time as evidenced by the release of Zigbee IP and is representative of how IEEE 802.15.4 can be leveraged at the PHY and MAC layers, independent of the protocol layers above. For these reasons, both Zigbee and Zigbee IP are discussed in more detail in the following sections.

ZigBee

Based on the idea of ZigBee-style networks in the late 1990s, the first ZigBee specification was ratified in 2004, shortly after the release of the IEEE 802.15.4 specification the previous year. While not released as a typical standard, like an RFC, ZigBee still had industry support from more than 100 companies upon its initial publication. This industry support has grown to more than 400 companies that are members of the ZigBee Alliance. Similar to the Wi-Fi Alliance, the Zigbee Alliance is an industry group formed to certify interoperability between vendors and it is committed to driving and evolving ZigBee as an IoT solution for interconnecting smart objects.

ZigBee solutions are aimed at smart objects and sensors that have low bandwidth and low power needs. Furthermore, products that are ZigBee compliant and certified by the ZigBee Alliance should interoperate even though different vendors may manufacture them.

The main areas where ZigBee is the most well-known include automation for commercial, retail, and home applications and smart energy. In the industrial and commercial automation space, ZigBee-based devices can handle various functions, from measuring temperature and humidity to tracking assets. For home automation, ZigBee can control lighting, thermostats, and security functions.

ZigBee Smart Energy brings together a variety of interoperable products, such as smart meters, that can monitor and control the use and delivery of utilities, such as electricity and water.

The traditional ZigBee stack is illustrated in Figure 4-3. As mentioned previously, ZigBee utilizes the IEEE 802.15.4 standard at the lower PHY and MAC layers. ZigBee specifies the network and security layer and application support layer that sit on top of the lower layers.

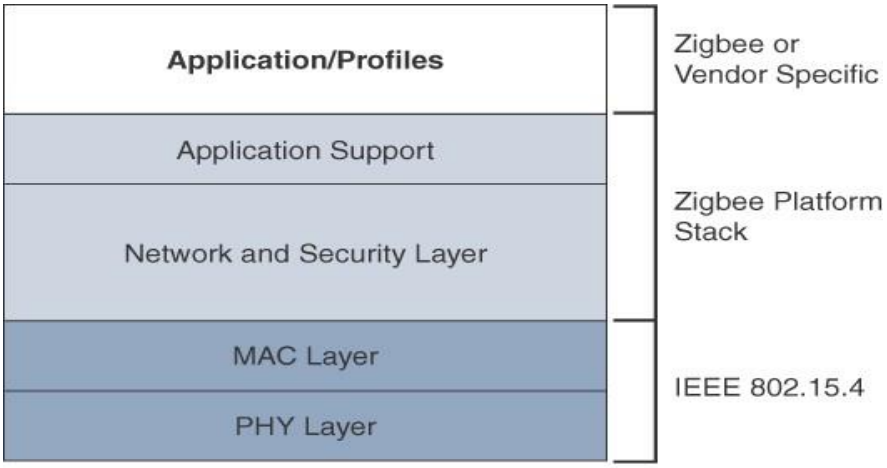


Figure 4-3 *High-Level ZigBee Protocol Stack*

The ZigBee network and security layer provides mechanisms for network startup, configuration, routing, and securing communications. This includes calculating routing paths in what is often a changing topology, discovering neighbors, and managing the routing tables as devices join for the first time. The network layer is also responsible for forming the appropriate topology, which is often a mesh but could be a star or tree as well. From a security perspective, ZigBee utilizes 802.15.4 for security at the MAC layer, using the Advanced Encryption Standard (AES) with a 128-bit key and also provides security at the network and application layers.

ZigBee IP

With the introduction of ZigBee IP, the support of IEEE 802.15.4 continues, but the IP and TCP/UDP protocols and various other open standards are now supported at the network and transport layers. The ZigBee-specific layers are now found only at the top of the protocol stack for the applications. ZigBee IP was created to embrace the open standards coming from the IETF's work on LLNs, such as IPv6, 6LoWPAN, and RPL. They provide for low-bandwidth, low-power, and cost- effective communications when connecting smart objects.

ZigBee IP is a critical part of the Smart Energy (SE) Profile 2.0 specification from the ZigBee Alliance. SE 2.0 is aimed at smart metering and residential energy management systems. In fact, ZigBee IP was designed specifically for SE 2.0 but it is not limited to this use case. Any other applications that need a standards-based IoT stack can utilize Zigbee IP. The ZigBee IP stack is shown in Figure 4-4.

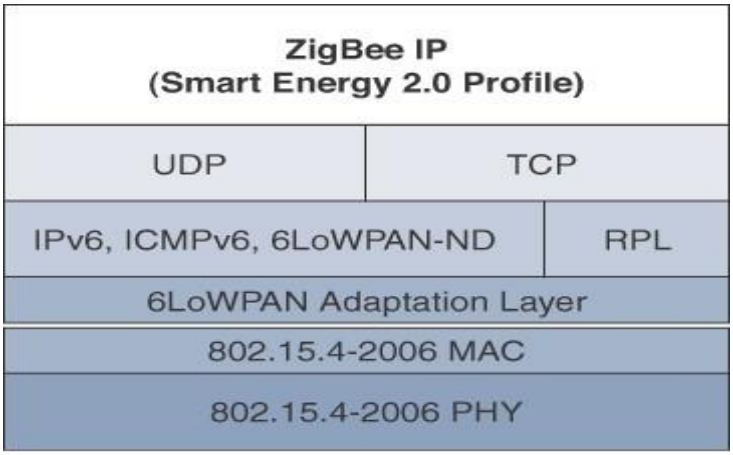


Figure 4-4 *ZigBee IP Protocol Stack*

ZigBee IP supports 6LoWPAN as an adaptation layer. The 6LoWPAN mesh addressing header is not required as ZigBee IP utilizes the mesh-over or route-over method for forwarding packets. ZigBee IP requires the support of 6LoWPAN's fragmentation and header compression schemes. At the network layer, all ZigBee IP nodes support IPv6, ICMPv6, and 6LoWPAN Neighbor Discovery (ND), and utilize RPL for the routing of packets across the mesh network. Both TCP and UDP are also supported, to provide both connection-oriented and connectionless service.

Physical Layer

The 802.15.4 standard supports an extensive number of PHY options that range from 2.4 GHz to sub-GHz frequencies in ISM bands. original IEEE 802.15.4-2003 standard specified only three PHY options based on **Direct Sequence Spread Spectrum (DSSS)** modulation. DSSS is a modulation technique in which a signal is intentionally spread in the frequency domain, resulting in greater bandwidth. The original physical layer transmission options were as follows:

- 2.4 GHz, 16 channels, with a data rate of 250 kbps
- 915 MHz, 10 channels, with a data rate of 40 kbps
- 868 MHz, 1 channel, with a data rate of 20 kbps

Note that only the 2.4 GHz band operates worldwide. The 915 MHz band operates mainly in North and South America, and the 868 MHz frequencies are used in Europe, the Middle East, and Africa. IEEE 802.15.4-2006, 802.15.4-2011, and IEEE 802.15.4-2015 introduced additional PHY communication options, including the following:

- **OQPHY:** This is DSSS PHY, employing offset quadrature phase-shift keying (OQPSK) modulation. OQPSK is a modulation technique that uses four unique bit values that are signaled by phase changes. An offset function that is present during phase shifts allows data to be transmitted more reliably.
- **BPSK PHY:** This is DSSS PHY, employing binary phase-shift keying (BPSK) modulation. BPSK specifies two unique phase shifts as its data encoding scheme.
- **ASK PHY:** This is parallel sequence spread spectrum (PSSS) PHY, employing amplitude shift keying (ASK) and BPSK modulation. PSSS is an advanced encoding scheme that offers increased range, throughput, data rates, and signal integrity compared to DSSS. ASK uses amplitude shifts instead of phase shifts to signal different bit values.

Figure 4-5 shows the frame for the 802.15.4 physical layer. The synchronization header for this frame is composed of the Preamble and the Start of Frame Delimiter fields. The Preamble field is a 32-bit 4-byte (for parallel construction) pattern that identifies the start of the frame and is used to synchronize the data transmission. The Start of Frame Delimiter field informs the receiver that frame contents start immediately after this byte.

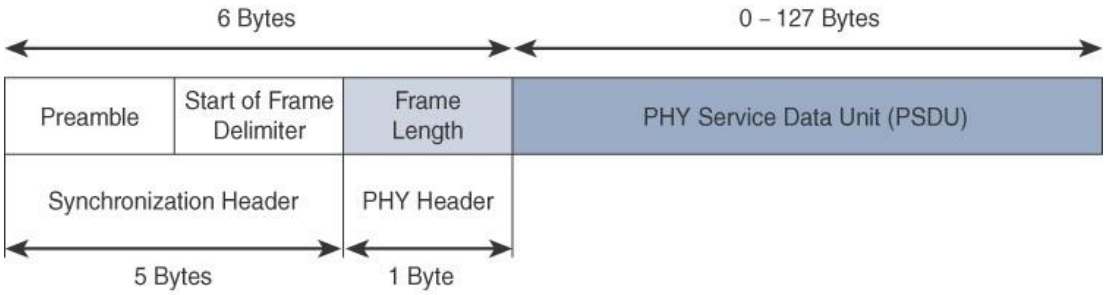


Figure 4-5 *IEEE 802.15.4 PHY Format*

The PHY Header portion of the PHY frame shown in Figure 4-5 is simply a frame length value. It lets the receiver know how much total data to expect in the PHY service data unit (PSDU) portion of the 802.4.15 PHY. The PSDU is the data field or payload.

MAC Layer

The IEEE 802.15.4 MAC layer manages access to the PHY channel by defining how devices in the same area will share the frequencies allocated. At this layer, the scheduling and routing of data frames are also coordinated. The 802.15.4 MAC layer performs the following tasks:

- Network beaconing for devices acting as coordinators (New devices use beacons to join an 802.15.4 network)
- PAN association and disassociation by a device
- Device security
- Reliable link communications between two peer MAC entities

The MAC layer achieves these tasks by using various predefined frame types. In fact, four types of MAC frames are specified in 802.15.4:

- **Data frame:** Handles all transfers of data
- **Beacon frame:** Used in the transmission of beacons from a PAN coordinator
- **Acknowledgement frame:** Confirms the successful reception of a frame
- **MAC command frame:** Responsible for control communication between devices

Each of these four 802.15.4 MAC frame types follows the frame format shown in Figure 4-6. In Figure 4-6, notice that the MAC frame is carried as the PHY payload. The 802.15.4 MAC frame can be broken down into the MAC Header, MAC Payload, and MAC Footer fields.

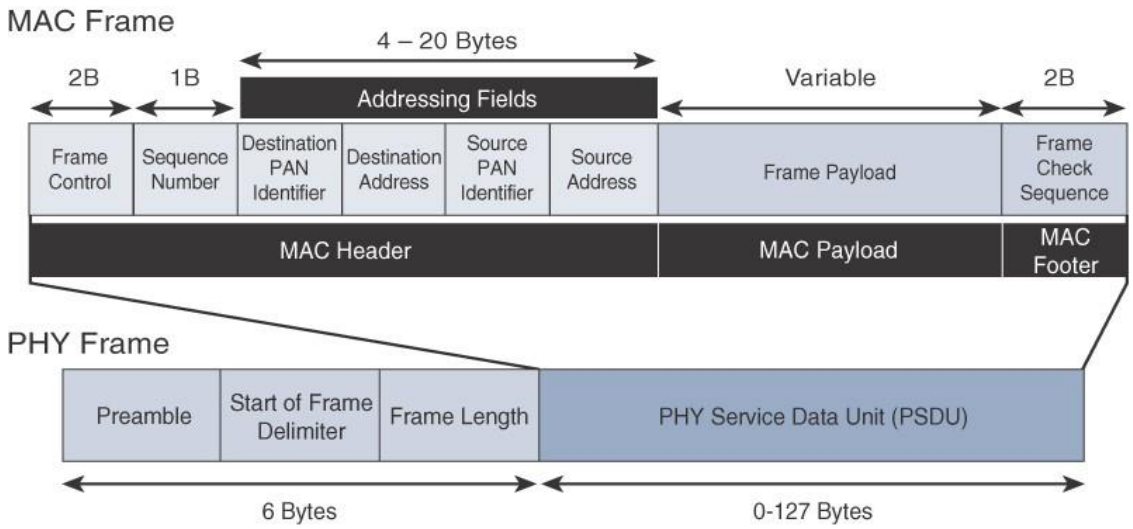


Figure 4-6 *IEEE 802.15.4 MAC Format*

- The MAC Header field is composed of the Frame Control, Sequence Number and the Addressing fields. The Frame Control field defines attributes such as frame type, addressing modes, and other control flags. The Sequence Number field indicates the sequence identifier for the frame. The Addressing field specifies the Source and Destination PAN Identifier fields as well as the Source and Destination Address fields.
- lThe MAC Payload field varies by individual frame type. For example, beacon frames have specific fields and payloads related to beacons, while MAC command frames have different fields present.
- The MAC Footer field is nothing more than a frame check sequence (FCS). An FCS is a calculation based on the data in the frame that is used by the receiving side to confirm the integrity of the data in the frame.

Topology

IEEE 802.15.4–based networks can be built as star, peer-to-peer, or mesh topologies. Mesh networks tie together many nodes. This allows nodes that would be out of range if trying to communicate directly to leverage intermediary nodes to transfer communications. Please note that every 802.15.4 PAN should be set up with a unique ID. All the nodes in the same 802.15.4 network should use the same PAN ID. Figure 4-7 shows an example of an 802.15.4 mesh network with a PAN ID of 1.

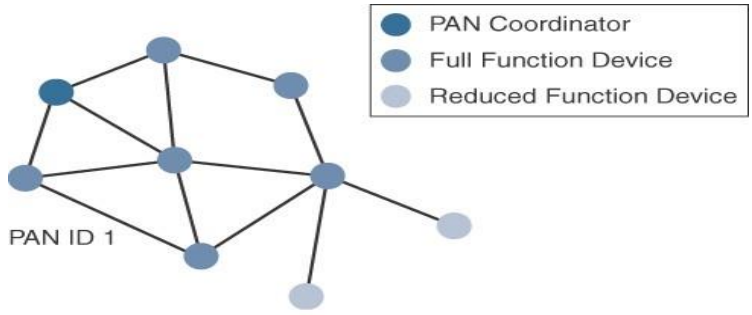


Figure 4-7 *802.15.4 Sample Mesh Network Topology*

A minimum of one FFD acting as a PAN coordinator is required to deliver services that allow other devices to associate and form a cell or PAN. Notice in Figure 4-7 that a single PAN coordinator is identified for PAN ID 1. FFD devices can communicate with any other devices, whereas RFD devices can communicate only with FFD devices.

Security

The IEEE 802.15.4 specification uses Advanced Encryption Standard (AES) with a 128-bit key length as the base encryption algorithm for securing its data. Established by the US National Institute of Standards and Technology in 2001, AES is a block cipher, which means it operates on fixed-size blocks of data. The use of AES by the US government and its widespread adoption in the private sector has helped it become one of the most popular algorithms used in symmetric key cryptography. (A symmetric key means that the same key is used for both the encryption and decryption of the data.) In addition to encrypting the data, AES in 802.15.4 also validates the data that is sent. This is accomplished by a message integrity code (MIC), which is calculated for the entire frame using the same AES key that is used for encryption.

Enabling these security features for 802.15.4 changes the frame format slightly and consumes some of the payload. Using the Security Enabled field in the Frame Control portion of the 802.15.4 header is the first step to enabling AES encryption. This field is a single bit that is set to 1 for security. Once this bit is set, a field called the Auxiliary Security Header is created after the Source Address field, by stealing some bytes from the Payload field. Figure 4-8 shows the IEEE 802.15.4 frame format at a high level, with the Security Enabled bit set and the Auxiliary Security Header field present.

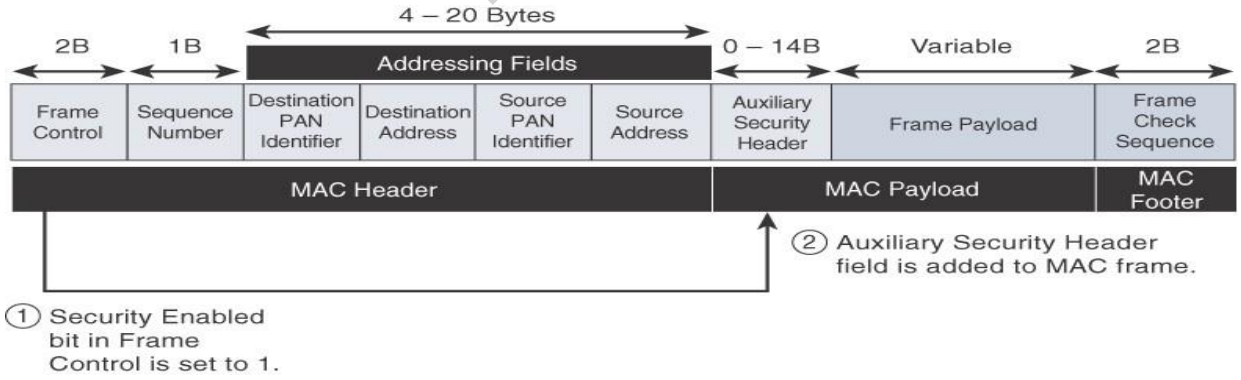


Figure 4-8 *Frame Format with the Auxiliary Security Header Field for 802.15.4-2006 and Later Versions*

Competitive Technologies

A competitive radio technology that is different in its PHY and MAC layers is DASH7. DASH7 was originally based on the ISO18000-7 standard and positioned for industrial communications, whereas IEEE 802.15.4 is more generic. Commonly employed in active radio frequency identification (RFID) implementations, DASH7 was used by US military forces for many years, mainly for logistics purposes. Active RFID utilizes radio waves generated by a battery-powered tag on an object to enable continuous tracking.

The current DASH7 technology offers low power consumption, a compact protocol stack, range up to 1 mile, and AES encryption. Frequencies of 433 MHz, 868 MHz, and 915 MHz have been defined, enabling data rates up to 166.667 kbps and a maximum payload of 256 bytes. DASH7 is promoted by the DASH7 Alliance, which has evolved the protocol from its active RFID niche into a wireless sensor network technology that is aimed at the commercial market.

3.6.2 IEEE 802.15.4g and 802.15.4e

The IEEE frequently makes amendments to the core 802.15.4 specification, before integrating them into the next revision of the core specification. When these amendments are made, a lowercase letter is appended. Two such examples of this are 802.15.4e-2012 and 802.15.4g-2012, both of which are especially relevant to the subject of IoT. Both of these amendments were integrated in IEEE 802.15.4-2015 but are often still referred to by their amendment names.

The IEEE 802.15.4e amendment of 802.15.4-2011 expands the MAC layer feature set to remedy the disadvantages associated with 802.15.4, including MAC reliability, unbounded latency, and multipath fading. In addition to making general enhancements to the MAC layer, IEEE 802.15.4e also made improvements to better cope with certain application domains, such as factory and process automation and smart grid.

IEEE 802.15.4g-2012 is also an amendment to the IEEE 802.15.4-2011 standard, and just like 802.15.4e-2012, it has been fully integrated into the core IEEE 802.15.4-2015 specification. The focus of this specification is the smart grid or, more specifically, smart utility network communication. 802.15.4g seeks to optimize large outdoor wireless mesh networks for field area networks (FANs). New PHY definitions are introduced, as well as some MAC modifications needed to support their implementation. This technology applies to IoT use cases such as the following:

- Distribution automation and industrial **supervisory control and data acquisition (SCADA)** environments for remote monitoring and control.
- Public lighting
- Image Environmental wireless sensors in smart cities
- Electrical vehicle charging stations
- Smart parking meters
- Microgrids
- Renewable energy

Standardization and Alliances

Because 802.15.4g-2012 and 802.15.4e-2012 are simply amendments to IEEE 802.15.4-2011, the same IEEE 802.15 Task Group 4 standards body authors, maintains, and integrates them into the next release of the core specification. However, the additional capabilities and options provided by 802.15.4g-2012 and 802.15.4e-2012 led to additional difficulty in achieving the interoperability between devices and mixed vendors that users requested.

To guarantee interoperability, the Wi-SUN Alliance was formed. (SUN stands for smart utility network.) This organization is not a standards body but is instead an industry alliance that defines communication profiles for smart utility and related networks. The Wi-SUN Alliance performs the same function as the Wi-Fi Alliance and WiMAX Forum. Each of these organizations has an associated standards body as well as a commercial name, as shown in Table 4-3.

Commercial Name/Trademark	Industry Organization	Standards Body
Wi-Fi	Wi-Fi Alliance	IEEE 802.11 Wireless LAN
WiMAX	WiMAX Forum	IEEE 802.16 Wireless MAN
Wi-SUN	Wi-SUN Alliance	IEEE 802.15.4g Wireless SUN

Table 4-3 *Industry Alliances for Some Common IEEE Standards*

Physical Layer

In IEEE 802.15.4g-2012, the original IEEE 802.15.4 maximum PSDU or payload size of 127 bytes was increased for the SUN PHY to 2047 bytes. This provides a better match for the greater packet sizes found in many upper-layer protocols. For example, the default IPv6 MTU setting is 1280 bytes. Fragmentation is no longer necessary at Layer 2 when IPv6 packets are transmitted over IEEE 802.15.4g MAC frames. Also, the error protection was improved in IEEE 802.15.4g by evolving the CRC from 16 to 32 bits.

The SUN PHY, as described in IEEE 802.15.4g-2012, supports multiple data rates in bands ranging from 169 MHz to 2.4 GHz. These bands are covered in the unlicensed ISM frequency spectrum specified by various countries and regions. Within these bands, data must be modulated onto the frequency using at least one of the following PHY mechanisms to be IEEE 802.15.4g compliant:

- **Multi-Rate and Multi-Regional Frequency Shift Keying (MR-FSK):** Offers good transmit power efficiency due to the constant envelope of the transmit signal
- **Multi-Rate and Multi-Regional Orthogonal Frequency Division Multiplexing (MR-OFDM):** Provides higher data rates but may be too complex for low-cost and low-power devices
- **Multi-Rate and Multi-Regional Offset Quadrature Phase-Shift Keying (MR-O-QPSK):** Shares the same characteristics of the IEEE 802.15.4-2006 O-QPSK PHY, making multi- mode systems more cost-effective and easier to design

Enhanced data rates and a greater number of channels for channel hopping are available, depending on the frequency bands and modulation. For example, for the 902–928 MHz ISM band that is used in the United States, MR-FSK provides 50, 150, or 200 kbps. MR-OFDM at this same frequency allows up to 800 kbps. Other frequencies provide their own settings.

MAC Layer

While the IEEE 802.15.4e-2012 amendment is not applicable to the PHY layer, it is pertinent to the MAC layer. This amendment enhances the MAC layer through various functions, which may be selectively enabled based on various implementations of the standard. In fact, if interoperability is a "must have," then using profiles defined by organizations such as Wi-SUN is necessary. The following are some of the main enhancements to the MAC layer proposed by IEEE 802.15.4e-2012:

- **Time-Slotted Channel Hopping (TSCH):** TSCH is an IEEE 802.15.4e-2012 MAC operation mode that works to guarantee media access and channel diversity. Channel hopping, also known as frequency hopping, utilizes different channels for transmission at different times. TSCH divides time into fixed time periods, or "time slots," which offer guaranteed bandwidth and predictable latency. In a time slot, one packet and its acknowledgement can be transmitted, increasing network capacity because multiple nodes can communicate in the same time slot, using different channels.
- **Information elements:** Information elements (IEs) allow for the exchange of information at the MAC layer in an extensible manner, either as header IEs (standardized) and/or payload IEs (private). Specified in a tag, length, value (TLV) format, the IE field allows frames to carry additional metadata to support MAC layer services. These services may include IEEE 802.15.9 key management, Wi-SUN 1.0 IEs to broadcast and unicast schedule timing information, and frequency hopping synchronization information for the 6TiSCH architecture.
- **Enhanced beacons (EBs):** EBs extend the flexibility of IEEE 802.15.4 beacons to allow the construction of application-specific beacon content. This is accomplished by including relevant IEs in EB frames. Some IEs that may be found in EBs include network metrics, frequency hopping broadcast schedule, and PAN information version.
- **Enhanced beacon requests (EBRs):** Like enhanced beacons, an enhanced beacon request (EBRs) also leverages IEs. The IEs in EBRs allow the sender to selectively specify the request of information. Beacon responses are then limited to what was requested in the EBR. For example, a device can query for a PAN that is allowing new devices to join or a PAN that supports a certain set of MAC/PHY capabilities.
- **Enhanced Acknowledgement:** The Enhanced Acknowledgement frame allows for the integration of a frame counter for the frame being acknowledged. This feature helps protect against certain attacks that occur when Acknowledgement frames are spoofed.

The 802.15.4e-2012 MAC amendment is quite often paired with the 802.15.4g-2012 PHY. Figure 4-9 details this frame format. Notice that the 802.15.4g-2012 PHY is similar to the 802.15.4 PHY in Figure 4-5. The main difference between the two is the payload size, with 802.15.4g supporting up to 2047 bytes and 802.15.4 supporting only 127 bytes.

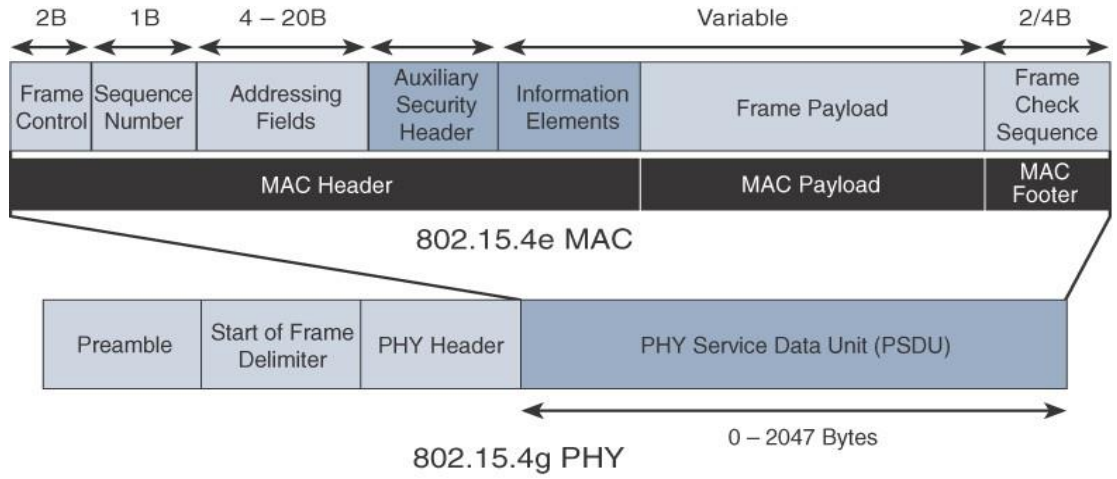


Figure 4-9 *IEEE 802.15.4g/e MAC Frame Format*

The 802.15.4e MAC is similar to the 802.15.4 MAC in Figure 4-6. The main changes shown in the IEEE 802.15.4e header in Figure 4-9 are the presence of the Auxiliary Security Header and Information Elements field.

- The Auxiliary Security header provides for the encryption of the data frame. This field is optionally supported in both 802.15.4e-2012 and 802.15.4, starting with the 802.15.4-2006 specification.
- The IE field contains one or more information elements that allow for additional information to be exchanged at the MAC layer.

Topology

Deployments of IEEE 802.15.4g-2012 are mostly based on a mesh topology. This is because a mesh topology is typically the best choice for use cases in the industrial and smart cities areas where 802.15.4g-2012 is applied. A mesh topology allows deployments to be done in urban or rural areas, expanding the distance between nodes that can relay the traffic of other nodes. Considering the use cases addressed by this technology, powered nodes have been the primary targets of implementations.

Security

Both IEEE 802.15.4g and 802.15.4e inherit their security attributes from the IEEE 802.15.4-2006 specification. Therefore, encryption is provided by AES, with a 128-bit key. In addition to the Auxiliary Security Header field initially defined in 802.15.4-2006, a secure acknowledgement and a secure Enhanced Beacon field complete the MAC layer security. Figure 4-10 shows a high-level overview of the security associated with an IEEE 802.15.4e MAC frame.

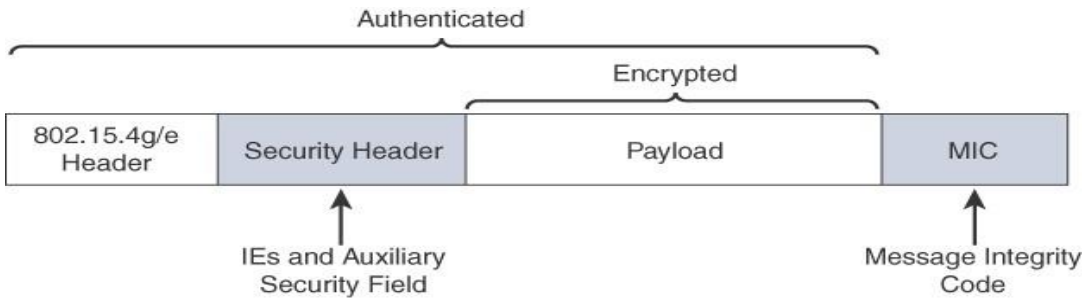


Figure 4-10 *IEEE 802.15.4g/e MAC Layer Security*

The full frame in Figure 4-10 gets authenticated through the MIC at the end of frame. The MIC is a unique value that is calculated based on the frame contents. The Security Header field denoted in Figure 4-10 is composed of the Auxiliary Security field and one or more Information Elements fields. Integration of the Information Elements fields allows for the adoption of additional security capabilities, such as the IEEE 802.15.9 Key Management Protocol (KMP) specification. KMP provides a means for establishing keys for robust datagram security. Without key management support, weak keys are often the result, leaving the security system open to attack.

Competitive Technologies

Competitive technologies to IEEE 802.15.4g and 802.15.4e parallel the technologies that also compete with IEEE 802.15.4, such as DASH7. In many ways, 802.15.4 and its various flavors of upper-layer protocols, as shown in Table 4-2, can be seen as competitors as well. IEEE 802.15.4 is well established and already deployed in many scenarios, mostly indoors.

3.6.3 IEEE 1901.2a

While most of the constrained network technologies relate to wireless, IEEE 1901.2a-2013 is a wired technology that is an update to the original IEEE 1901.2 specification. This is a standard for Narrowband Power Line Communication (NB-PLC). NB-PLC is a narrowband spectrum for low power, long range, and resistance to interference over the same wires that carry electric power. NB-PLC is often found in use cases such as the following

- **Smart metering:** NB-PLC can be used to automate the reading of utility meters, such as electric, gas, and water meters. This is true particularly in Europe, where PLC is the preferred technology for utilities deploying smart meter solutions.
- **Distribution automation:** NB-PLC can be used for distribution automation, which involves monitoring and controlling all the devices in the power grid.
- **Public lighting:** A common use for NB-PLC is with public lighting—the lights found in cities and along streets, highways, and public areas such as parks.
- **Electric vehicle charging stations:** NB-PLC can be used for electric vehicle charging stations, where the batteries of electric vehicles can be recharged.
- **Microgrids:** NB-PLC can be used for microgrids, local energy grids that can disconnect from the traditional grid and operate independently.
- **Renewable energy:** NB-PLC can be used in renewable energy applications, such as solar, wind power, hydroelectric, and geothermal heat.

All these use cases require a direct connection to the power grid. So it makes sense to transport IoT data across power grid connections that are already in place.

Standardization and Alliances

The first generations of NB-PLC implementations have generated a lot of interest from utilities in Europe but have often suffered from poor reliability, low throughput (in the range of a few hundred bits per second to a maximum of 2 kbps), lack of manageability, and poor interoperability. This has led several organizations (including standards bodies and alliance consortiums) to develop their own specifications for new generations of NB-PLC technologies. Most recent NB-PLC standards are based on orthogonal frequency-division multiplexing (OFDM). However, different standards from various vendors competing with one another have created a fragmented market. OFDM encodes digital data

on multiple carrier frequencies. This provides several parallel streams that suffer less from high frequency attenuation in copper wire and narrowband interference.

The HomePlug Alliance was one of the main industry organizations that drove the promotion and certification of PLC technologies, with IEEE 1901.2a being part of its HomePlug Netricity program. In 2016, the HomePlug Alliance made the decision to offer the alliance's broadband power line networking technology to a broader audience by making its technical specifications publicly available. It has also partnered with other alliances on continuing ongoing work. The HomePlug Alliance has struck a liaison agreement with the Wi-SUN Alliance with the goal of enabling hybrid smart grid networks that support both wireless and power line–wired connectivity. For more information on the HomePlug Alliance and Netricity, see www.homeplug.org.

Physical Layer

NB-PLC is defined for frequency bands from 3 to 500 kHz. Much as with wireless sub-GHz frequency bands, regional regulations and definitions apply to NB-PLC. The IEEE 1901.2 working group has integrated support for all world regions in order to develop a worldwide standard. Specifications include support for CENELEC A and B bands, US FCC-Low and FCC-above- CENELEC, and Japan ARIB bands. CENELEC is the French Comité Européen de Normalisation Électrotechnique, which in English translates to European Committee for Electrotechnical Standardization. This organization is responsible for standardization in the area of electrical engineering for Europe. The CENELEC A and B bands refer to 9–95 kHz and 95–125 kHz, respectively.

The FCC is the Federal Communications Commission, a US government organization that regulates interstate and international communications by radio, television, wire, satellite, and cable. The FCC-Low band encompasses 37.5–117.1875 kHz, and the FCC-above-CENELEC band is 154.6875–487.5 kHz. The FCC-above-CENELEC band may become the most useful frequency due to its higher throughput and reduced interference.

Figure 4-11 shows the various frequency bands for NB-PLC. Notice that the most well-known bands are regulated by CENELEC and the FCC, but the Japan Association of Radio Industries and Businesses (ARIB) band is also present. The two ARIB frequency bands are ARIB 1, 37.5–117.1875 kHz, and ARIB 2, 154.6875–403.125 kHz.

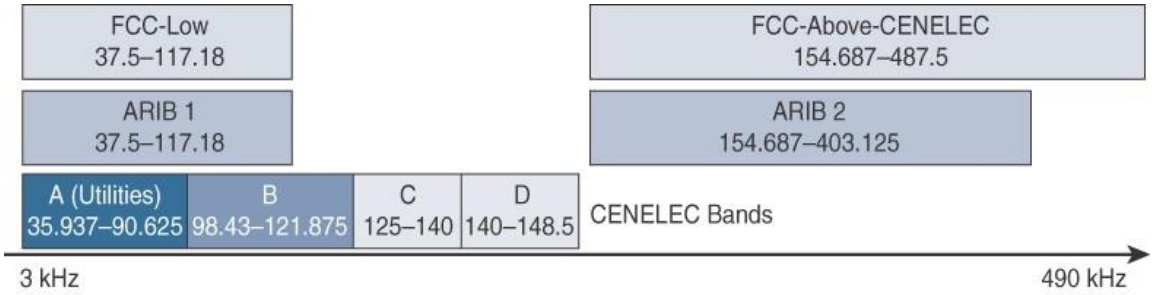


Figure 4-11 *NB-PLC Frequency Bands*

Based on OFDM, the IEEE 1901.2 specification leverages the best from other NB-PLC OFDM technologies that were developed previously. Therefore, IEEE 1901.2a supports the largest set of coding and enables both robustness and throughput. The standard includes tone maps and modulations,

such as robust modulation (ROBO), differential binary phase shift keying (DBPSK), differential quadrature phase shift keying (DQPSK), differential 8-point phase shift keying (D8PSK) for all bands, and optionally 16 quadrature amplitude modulation (16QAM) for some bands. ROBO mode transmits redundant information on multiple carriers, and DBPSK, DQPSK, and D8PSK are all variations of phase shift keying, where the phase of a signal is changed to signal a binary data transmission. ROBO utilizes QPSK modulation, and its throughput depends on the degree to which coding is repeated across streams. For example, standard ROBO uses a repetition of 4, and Super- ROBO utilizes a repetition of 6.

With IEEE 1901.2a, the data throughput rate has the ability to dynamically change, depending on the modulation type and tone map. For CENELEC A band, the data rate ranges from 4.5 kbps in ROBO mode to 46 kbps with D8PSK modulation. For the FCC-above-CENELEC frequencies, throughput varies from 21 kbps in ROBO mode to a maximum of 234 kbps using D8PSK.

One major difference between IEEE 802.15.4g/e and IEEE 1901.2a is the full integration of different types of modulation and tone maps by a single PHY layer in the IEEE 1901.2a specification. IEEE 802.15.4g/e doesn't really define a multi-PHY management algorithm. The PHY payload size can change dynamically, based on channel conditions in IEEE 1901.2a. Therefore, MAC sublayer segmentation is implemented. If the size of the MAC payload is too large to fit within one PHY service data unit (PSDU), the MAC payload is partitioned into smaller segments. MAC payload segmentation is done by dividing the MAC payload into multiple smaller amounts of data (segments), based on PSDU size. The segmentation may require the addition of padding bytes to the last payload segment so that the final MPDU fills the PSDU. All forms of addressing (unicast and broadcast) are subject to segmentation.

MAC Layer

The MAC frame format of IEEE 1901.2a is based on the IEEE 802.15.4 MAC frame but integrates the latest IEEE 802.15.4e-2012 amendment, which enables key features to be supported. (For more information on the 802.15.4 MAC frame format, refer to Figure 4-6. For the 802.15.4e MAC frame format, see Figure 4-9.) One of the key components brought from 802.15.4e to IEEE 1901.2a is information elements. With IE support, additional capabilities, such as IEEE 802.15.9 Key Management Protocol and SSID, are supported. Figure 4-12 provides an overview of the general MAC frame format for IEEE 1901.2. Note that the numeric value above each field in the frame shows the size of the field, in bytes.

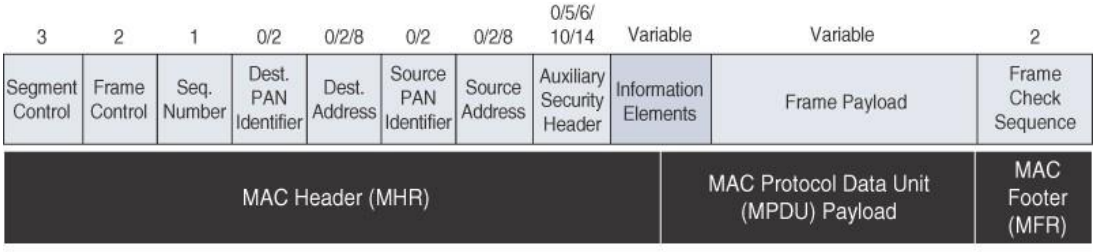


Figure 4-12 *General MAC Frame Format for IEEE 1901.2*

As shown in Figure 4-12, IEEE 1901.2 has a Segment Control field. This is a new field that was not present in our previous discussions of the MAC frame for 802.15.4 and 802.15.4e. This field handles

the segmentation or fragmentation of upper-layer packets with sizes larger than what can be carried in the MAC protocol data unit (MPDU).

Topology

Use cases and deployment topologies for IEEE 1901.2a are tied to the physical power lines. As with wireless technologies, signal propagation is limited by factors such as noise, interference, distortion, and attenuation. These factors become more prevalent with distance, so most NB-PLC deployments use some sort of mesh topology. Mesh networks offer the advantage of devices relaying the traffic of other devices so longer distances can be segmented. The IEEE 1901.2a standard offers the flexibility to run any upper-layer protocol. So, implementations of IPv6 6LoWPAN and RPL IPv6 protocols are supported.

Security

IEEE 1901.2a security offers similar features to IEEE 802.15.4g. Encryption and authentication are performed using AES. In addition, IEEE 1901.2a aligns with 802.15.4g in its ability to support the IEEE 802.15.9 Key Management Protocol. However, some differences exist. These differences are mostly tied to the PHY layer fragmentation capabilities of IEEE 1901.2a and include the following:

- The Security Enabled bit in the Frame Control field should be set in all MAC frames carrying segments of an encrypted frame.
- If data encryption is required, it should be done before packet segmentation. During packet encryption, the Segment Control field should not be included in the input to the encryption algorithm.
- On the receiver side, the data decryption is done after packet reassembly.
- When security is enabled, the MAC payload is composed of the ciphered payload and the message integrity code (MIC) authentication tag for non-segmented payloads. If the payload is segmented, the MIC is part of the last packet (segment) only. The MIC authentication is computed using only information from the MHR of the frame carrying the first segment.

Competitive Technologies

In the domain of NB-PLC, two technologies compete against IEEE 1901.2a: G3-PLC (now ITU G.9903) and PRIME (now ITU G.9904). Both of these technologies were initially developed to address a single use case: smart metering deployment in Europe over the CENELEC A band. IEEE 1901.2a has portions of G3-PLC and PRIME, and it also competes with them. More specifically, G3- PLC is really close to IEEE 1901.2. The main differences include the fact that G3-PLC mandates data link layer protocol options for bootstrapping and allocating device addresses, and it is incompatible with IEEE 802.15.4g/e and an end-to-end IPv6 model.

3.6.4 IEEE 802.11ah

In unconstrained networks, IEEE 802.11 Wi-Fi is certainly the most successfully deployed wireless technology. This standard is a key IoT wireless access technology, either for connecting endpoints such as fog computing nodes, high-data-rate sensors, and audio or video analytics devices or for deploying Wi-Fi backhaul infrastructures, such as outdoor Wi-Fi mesh in smart cities, oil and mining,

or other environments. However, Wi-Fi lacks sub-GHz support for better signal penetration, low power for battery-powered nodes, and the ability to support a large number of devices. For these reasons, the IEEE 802.11 working group launched a task group named IEEE 802.11ah to specify a sub-GHz version of Wi-Fi. Three main use cases are identified for IEEE 802.11ah:

- **Sensors and meters covering a smart grid:** Meter to pole, environmental/agricultural monitoring, industrial process sensors, indoor healthcare system and fitness sensors, home and building automation sensors.

  Backhaul aggregation of industrial sensors and meter data: Potentially connecting IEEE 802.15.4g subnetworks,
- **Extended range Wi-Fi:** For outdoor extended-range hotspot or cellular traffic offloading when distances already covered by IEEE 802.11a/b/g/n/ac are not good enough.

Standardization and Alliances

For the 802.11ah standard, the Wi-Fi Alliance defined a new brand called Wi-Fi HaLow. This marketing name is based on a play on words between "11ah" in reverse and "low power." It is similar to the word "hello" but it is pronounced "hay-low." The HaLow brand exclusively covers IEEE 802.11ah for sub-GHz device certification. Wi-Fi HaLow is a commercial designation for products incorporating IEEE 802.11ah technology.

Physical Layer

IEEE 802.11ah essentially provides an additional 802.11 physical layer operating in unlicensed sub-GHz bands. For example, various countries and regions use the following bands for IEEE 802.11ah: 868–868.6 MHz for EMEAR, 902–928 MHz and associated subsets for North America and Asia-Pacific regions, and 314–316 MHz, 430–434 MHz, 470–510 MHz, and 779–787 MHz for China.

Based on OFDM modulation, IEEE 802.11ah uses channels of 2, 4, 8, or 16 MHz (and also 1 MHz for low-bandwidth transmission). This is one-tenth of the IEEE 802.11ac channels, resulting in one- tenth of the corresponding data rates of IEEE 802.11ac. The IEEE 802.11ac standard is a high-speed wireless LAN protocol at the 5 GHz band that is capable of speeds up to 1 Gbps. While 802.11ah does not approach this transmission speed (as it uses one-tenth of 802.11ac channel width, it reaches one-tenth of 802.11ac speed), it does provide an extended range for its lower speed data. For example, at a data rate of 100 kbps, the outdoor transmission range for IEEE 802.11ah is expected to be 0.62 mile.

MAC Layer

The IEEE 802.11ah MAC layer is optimized to support the new sub-GHz Wi-Fi PHY while providing low power consumption and the ability to support a larger number of endpoints. Enhancements and features specified by IEEE 802.11ah for the MAC layer include the following:

- **Number of devices:** Has been scaled up to 8192 per access point.
- **MAC header:** Has been shortened to allow more efficient communication.
- **Null data packet (NDP) support:** Is extended to cover several control and management frames. Relevant information is concentrated in the PHY header and the additional overhead associated with decoding the MAC header and data payload is avoided. This change makes the control frame exchanges efficient and less power-consuming for the receiving stations.
- **Grouping and sectorization:** Enables an AP to use sector antennas and also group stations (distributing a group ID). In combination with RAW and TWT, this mechanism reduces

contention in large cells with many clients by restricting which group, in which sector, can contend during which time window. (Sectors are described in more detail in the following section.)

- **Restricted access window (RAW):** Is a control algorithm that avoids simultaneous transmissions when many devices are present and provides fair access to the wireless network. By providing more efficient access to the medium, additional power savings for battery-powered devices can be achieved, and collisions are reduced.
- **Target wake time (TWT):** Reduces energy consumption by permitting an access point to define times when a device can access the network. This allows devices to enter a low-power state until their TWT time arrives. It also reduces the probability of collisions in large cells with many clients.
- **Speed frame exchange:** Enables an AP and endpoint to exchange frames during a reserved transmit opportunity (TXOP). This reduces contention on the medium, minimizes the number of frame exchanges to improve channel efficiency, and extends battery life by keeping awake times short.

From the above feature list the 802.11ah MAC layer is focused on power consumption and mechanisms to allow low-power Wi-Fi stations to wake up less often and operate more efficiently. This sort of MAC layer is ideal for IoT devices that often produce short, low-bit-rate transmissions.

Topology

While IEEE 802.11ah is deployed as a star topology, it includes a simple hops relay operation to extend its range. This relay option is not capped, but the IEEE 802.11ah task group worked on the assumption of two hops. It allows one 802.11ah device to act as an intermediary and relay data to another. In some ways, this is similar to a mesh, and it is important to note that the clients and not the access point handle the relay function. This relay operation can be combined with a higher transmission rate or modulation and coding scheme (MCS). This means that a higher transmit rate is used by relay devices talking directly to the access point.

Sectorization is a technique that involves partitioning the coverage area into several sectors to get reduced contention within a certain sector. This technique is useful for limiting collisions in cells that have many clients. This technique is also often necessary when the coverage area of 802.11ah access points is large, and interference from neighboring access points is problematic. Sectorization uses an antenna array and beam-forming techniques to partition the cell-coverage area. Figure 4-14 shows an example of 802.11ah sectorization.

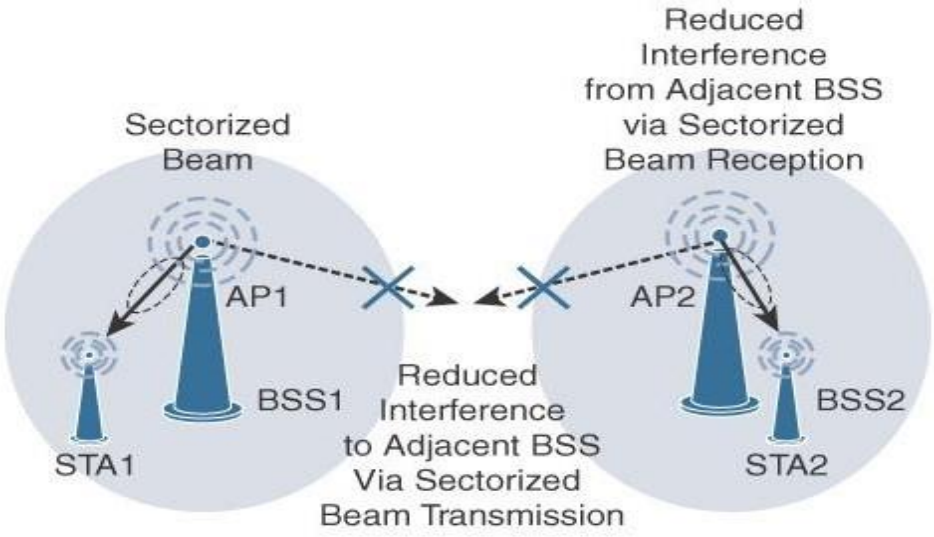


Figure 4-14 *IEEE 802.11ah Sectorization*

Security

No additional security has been identified for IEEE 802.11ah compared to other IEEE 802.11 specifications. These protocols include IEEE 802.15.4, IEEE 802.15.4e, and IEEE 1901.2a, and the security information for them is also applicable to IEEE 802.11ah.

Competitive Technologies

Competitive technologies to IEEE 802.11ah are IEEE 802.15.4 and IEEE 802.15.4e, along with the competitive technologies highlighted in each of their sections.

3.6.5 LoRaWAN

In recent years, a new set of wireless technologies known as Low-Power Wide-Area (LPWA) has received a lot of attention from the industry and press. Particularly well adapted for long-range and battery-powered endpoints, LPWA technologies open new business opportunities to both services providers and enterprises considering IoT solutions. This section discusses an example of an unlicensed-band LPWA technology, known as LoRaWAN, and the next section, "NB-IoT and Other LTE Variations," reviews licensed-band alternatives from the 3rd Generation Partnership Project (3GPP).

Standardization and Alliances

Initially, LoRa was a physical layer, or Layer 1, modulation that was developed by a French company named Cycleo. Later, Cycleo was acquired by Semtech. Optimized for long-range, two- way communications and low power consumption, the technology evolved from Layer 1 to a broader scope through the creation of the LoRa Alliance. It quickly achieved industry support and currently has hundreds of members.

Semtech LoRa as a Layer 1 PHY modulation technology is available through multiple chipset vendors. To differentiate from the physical layer modulation known as LoRa, the LoRa Alliance uses

the term LoRaWAN to refer to its architecture and its specifications that describe end-to-end LoRaWAN communications and protocols. Figure 4-15 provides a high-level overview of the LoRaWAN layers. In this figure, notice that Semtech is responsible for the PHY layer, while the LoRa Alliance handles the MAC layer and regional frequency bands.

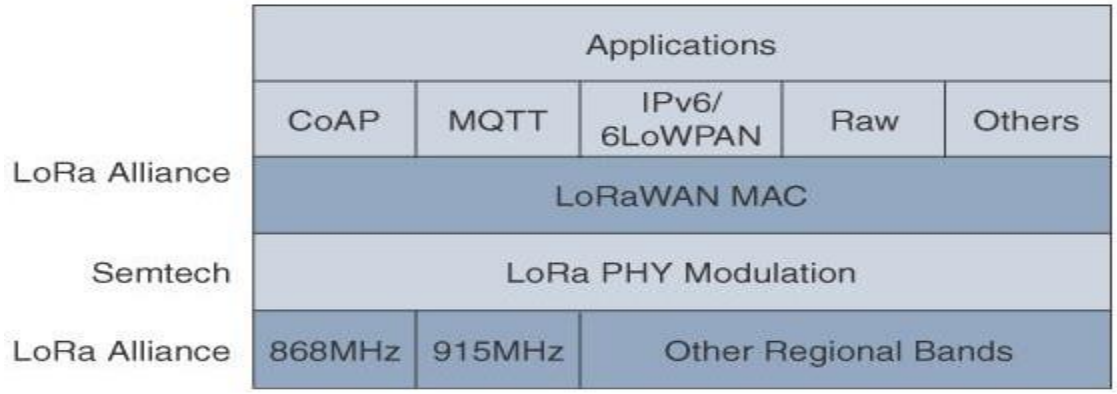


Figure 4-15 *LoRaWAN Layers*

Overall, the LoRa Alliance owns and manages the roadmap and technical development of the LoRaWAN architecture and protocol. This alliance also handles the LoRaWAN endpoint certification program and technology promotion through its certification and marketing committees.

Physical Layer

Semtech LoRa modulation is based on chirp spread spectrum modulation, which trades a lower data rate for receiver sensitivity to significantly increase the communication distance. In addition, it allows demodulation below the noise floor, offers robustness to noise and interference, and manages a single channel occupation by different spreading factors. This enables LoRa devices to receive on multiple channels in parallel.

LoRaWAN 1.0.2 regional specifications describe the use of the main unlicensed sub-GHz frequency bands of 433 MHz, 779–787 MHz, 863–870 MHz, and 902–928 MHz, as well as regional profiles for a subset of the 902–928 MHz bandwidth. For example, Australia utilizes 915–928 MHz frequency bands, while South Korea uses 920–923 MHz and Japan uses 920–928 MHz.

The data rate in LoRaWAN varies depending on the frequency bands and adaptive data rate (ADR). ADR is an algorithm that manages the data rate and radio signal for each endpoint. The ADR algorithm ensures that packets are delivered at the best data rate possible and that network performance is both optimal and scalable. Endpoints close to the gateways with good signal values transmit with the highest data rate, which enables a shorter transmission time over the wireless network, and the lowest transmit power. Meanwhile, endpoints at the edge of the link budget communicate at the lowest data rate and highest transmit power.

An important feature of LoRa is its ability to handle various data rates via the spreading factor. Devices with a low spreading factor (SF) achieve less distance in their communications but transmit at faster speeds, resulting in less airtime. A higher SF provides slower transmission rates but achieves a higher reliability at longer distances. Table 4-4 illustrates how LoRaWAN data rates can vary depending on the associated spreading factor for the two main frequency bands, 863–870 MHz and 902–928 MHz.

Configuration	863–870 MHz bps	902–928 MHz bps
LoRa: SF12/125 kHz	250	N/A
LoRa: SF11/125 kHz	440	N/A
LoRa: SF10/125 kHz	980	980
LoRa: SF9/125 kHz	1760	1760
LoRa: SF8/125 kHz	3125	3125
LoRa: SF7/125 kHz	5470	5470
LoRa: SF7/250 kHz	11,000	N/A
FSK: 50 kbps	50,000	N/A
LoRa: SF12/500 kHz	N/A	980
LoRa: SF11/500 kHz	N/A	1760
LoRa: SF10/500 kHz	N/A	3900
LoRa: SF9/500 kHz	N/A	7000
LoRa: SF8/500 kHz	N/A	12,500
LoRa: SF7/500 kHz	N/A	21,900

Table 4-4 *LoRaWAN Data Rate Example*

In Table 4-4, notice the relationship between SF and data rate. For example, at an SF value of 12 for 125 kHz of channel bandwidth, the data rate is 250 bps. However, when the SF is decreased to a value of 7, the data rate increases to 5470 bps. Channel bandwidth values of 125 kHz, 250 kHz, and 500 kHz are also evident in Table 4-4. The effect of increasing the bandwidth is that faster data rates can be achieved for the same spreading factor.

MAC Layer

As mentioned previously, the MAC layer is defined in the LoRaWAN specification. This layer takes advantage of the LoRa physical layer and classifies LoRaWAN endpoints to optimize their battery life and ensure downstream communications to the LoRaWAN endpoints. The LoRaWAN specification documents three classes of LoRaWAN devices:

- **Class A:** This class is the default implementation. Optimized for battery-powered nodes, it allows bidirectional communications, where a given node is able to receive downstream traffic after transmitting. Two receive windows are available after each transmission.
- **Class B:** This class was designated "experimental" in LoRaWAN 1.0.1 until it can be better defined. A Class B node or endpoint should get additional receive windows compared to Class A, but gateways must be synchronized through a beaconing process.
- **Class C:** This class is particularly adapted for powered nodes. This classification enables a node to be continuously listening by keeping its receive window open when not transmitting.

LoRaWAN messages, either uplink or downlink, have a PHY payload composed of a 1-byte MAC header, a variable-byte MAC payload, and a MIC that is 4 bytes in length. The MAC payload size depends on the frequency band and the data rate, ranging from 59 to 230 bytes for the 863–870 MHz band and 19 to 250 bytes for the 902–928 MHz band. Figure 4-16 shows a high-level LoRaWAN MAC frame format.

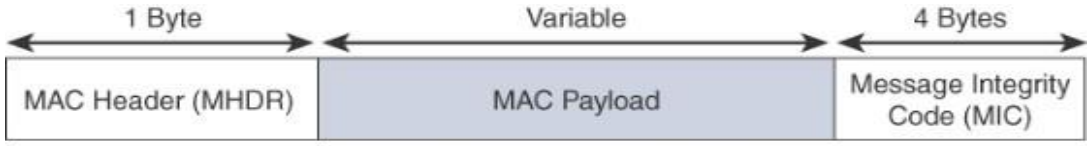


Figure 4-16 *High-Level LoRaWAN MAC Frame Format*

In version 1.0.x, LoRaWAN utilizes six MAC message types. LoRaWAN devices use join request and join accept messages for over-the-air (OTA) activation and joining the network. The other message types are unconfirmed data up/down and confirmed data up/down. A "confirmed" message is one that must be acknowledged, and "unconfirmed" signifies that the end device does not need to acknowledge. "up/down" is simply a directional notation identifying whether the message flows in the uplink or downlink path. Uplink messages are sent from endpoints to the network server and are relayed by one or more LoRaWAN gateways. Downlink messages flow from the network server to a single endpoint and are relayed by only a single gateway. Multicast over LoRaWAN is being considered for future versions.

LoRaWAN endpoints are uniquely addressable through a variety of methods, including the following:

- An endpoint can have a global end device ID or DevEUI represented as an IEEE EUI-64 address.
- An endpoint can have a global application ID or AppEUI represented as an IEEE EUI-64 address that uniquely identifies the application provider, such as the owner, of the end device.
- In a LoRaWAN network, endpoints are also known by their end device address, known as a DevAddr, a 32-bit address. The 7 most significant bits are the network identifier (NwkID), which identifies the LoRaWAN network. The 25 least significant bits are used as the network address (NwkAddr) to identify the endpoint in the network.

Topology

LoRaWAN topology is often described as a "star of stars" topology. As shown in Figure 4-17, the infrastructure consists of endpoints exchanging packets through gateways acting as bridges, with a central LoRaWAN network server. Gateways connect to the backend network using standard IP connections, and endpoints communicate directly with one or more gateways.

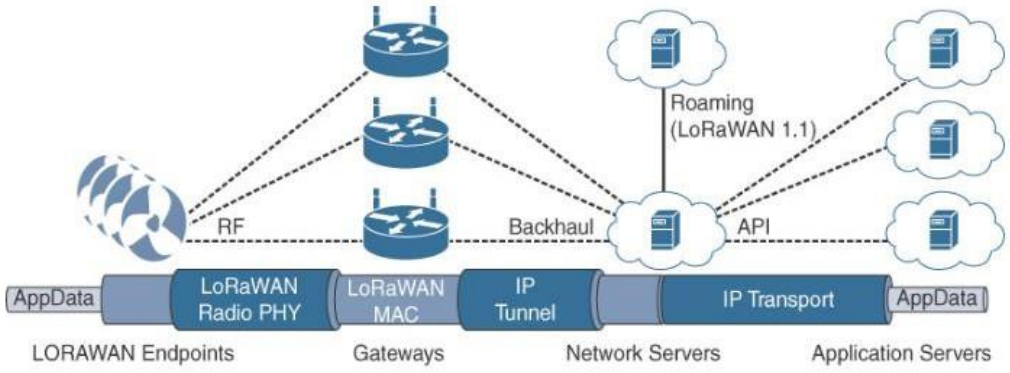


Figure 4-17 *LoRaWAN Architecture*

In Figure 4-17, LoRaWAN endpoints transport their selected application data over the LoRaWAN MAC layer on top of one of the supported PHY layer frequency bands. The application data is contained in upper protocol layers. These upper layers are not the responsibility of the LoRa Alliance, but best practices may be developed and recommended. These upper layers could just be raw data on top of the LoRaWAN MAC layer, or the data could be stacked in multiple protocols. Figure 4-17 also shows how LoRaWAN gateways act as bridges that relay between endpoints and the network servers. Multiple gateways can receive and transport the same packets. When duplicate packets are received, de-duplication is a function of the network server.

The LoRaWAN network server manages the data rate and radio frequency (RF) of each endpoint through the adaptive data rate (ADR) algorithm. ADR is a key component of the network scalability, performance, and battery life of the endpoints. The LoRaWAN network server forwards application data to the application servers, as depicted in Figure 4-17.

In future versions of the LoRaWAN specification, roaming capabilities between LoRaWAN network servers will be added. These capabilities will enable mobile endpoints to connect and roam between different LoRaWAN network infrastructures.

Security

Security in a LoRaWAN deployment applies to different components of the architecture, as detailed in Figure 4-18. LoRaWAN endpoints must implement two layers of security, protecting communications and data privacy across the network.

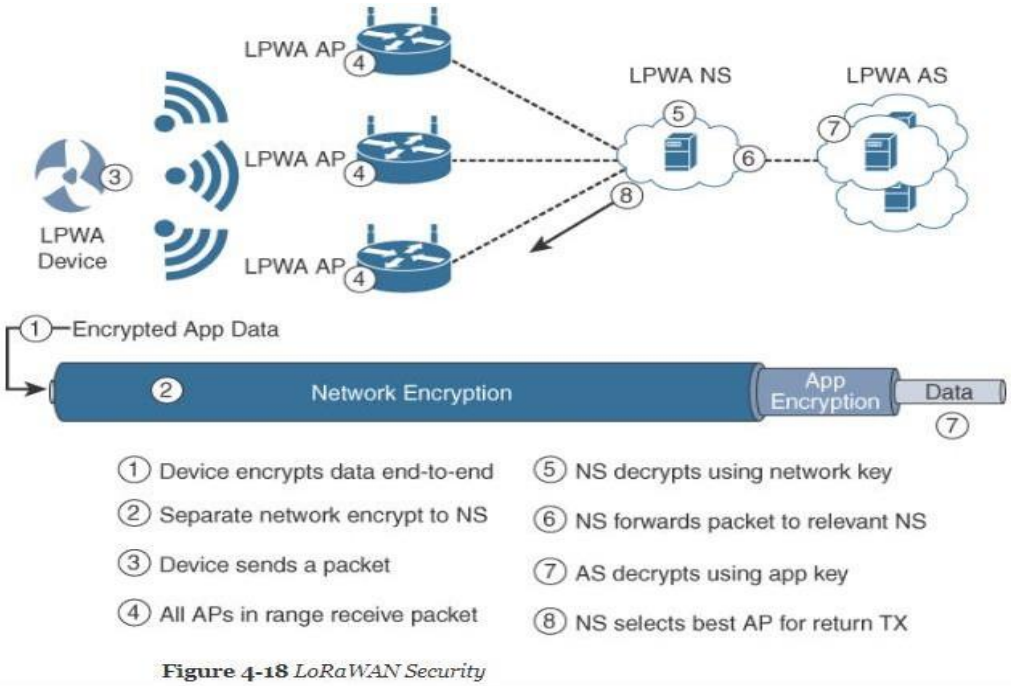


Figure 4-18 *LoRaWAN Security*

The first layer, called "network security" but applied at the MAC layer, guarantees the authentication of the endpoints by the LoRaWAN network server. Also, it protects LoRaWAN packets by performing encryption based on AES. Each endpoint implements a network session key (NwkSKey), used by both itself and the LoRaWAN network server. The NwkSKey ensures data integrity through computing and checking the MIC of every data message as well as encrypting and decrypting MAC-only data message payloads.

The second layer is an application session key (AppSKey), which performs encryption and decryption functions between the endpoint and its application server. Furthermore, it computes and checks the application-level MIC, if included. This ensures that the LoRaWAN service provider does not have access to the application payload if it is not allowed that access.

Endpoints receive their AES-128 application key (AppKey) from the application owner. This key is most likely derived from an application-specific root key exclusively known to and under the control of the application provider. For production deployments, it is expected that the LoRaWAN gateways are protected as well, for both the LoRaWAN traffic and the network management and operations over their backhaul link(s). This can be done using traditional VPN and IPsec technologies that demonstrate scaling in traditional IT deployments.

LoRaWAN endpoints attached to a LoRaWAN network must get registered and authenticated. This can be achieved through one of the two join mechanisms:

- **Activation by personalization (ABP):** Endpoints don't need to run a join procedure as their individual details, including DevAddr and the NwkSKey and AppSKey session keys, are

preconfigured and stored in the end device. This same information is registered in the LoRaWAN network server.

- **Over-the-air activation (OTAA):** Endpoints are allowed to dynamically join a particular LoRaWAN network after successfully going through a join procedure. The join procedure must be done every time a session context is renewed. During the join process, which involves the sending and receiving of MAC layer join request and join accept messages, the node establishes its credentials with a LoRaWAN network server, exchanging its globally unique DevEUI, AppEUI, and AppKey. The AppKey is then used to derive the session NwkSKey and AppSKey keys.

Competitive Technologies

LPWA solutions and technologies are split between unlicensed and licensed bands. The licensed- band technologies are dedicated to mobile service providers that have acquired spectrum licenses; they are discussed in the next section. In addition, several technologies are targeting the unlicensed- band LPWA market to compete against LoRaWAN. The LPWA market is quickly evolving. Table 4- 5 evaluates two of the best-established vendors known to provide LPWA options.

Characteristic	LoRaWAN	Sigfox	Ingenu Onramp
Frequency bands	433 MHz, 868 MHz, 902–928 MHz	433 MHz, 868 MHz, 902–928 MHz	2.4 GHz
Modulation	Chirp spread spectrum	Ultra-narrowband	DSSS
Topology	Star of stars	Star	Star; tree supported with an RPMA extender
Data rate	250 bps–50 kbps (868 MHz) 980 bps–21.9 kbps (915 MHz)	100 bps (868 MHz) 600 bps (915 MHz)	6 kbps
Adaptive data rate	Yes	No	No
Payload	59–230 bytes (868 MHz) 19–250 bytes (915 MHz)	12 bytes	6 bytes–10 KB
Two-way communications	Yes	Partial	Yes
Geolocation	Yes (LoRa GW version 2 reference design)	No	No
Roaming	Yes (LoRaWAN 1.1)	No	Yes
Specifications	LoRA Alliance	Proprietary	Proprietary

Table 4-5 *Unlicensed LPWA Technology Comparison*

3.6.6 NB-IoT and Other LTE Variations

Existing cellular technologies, such as GPRS, Edge, 3G, and 4G/LTE, are not particularly well adapted to battery-powered devices and small objects specifically developed for the Internet of Things. While industry players have been developing unlicensed-band LPWA technologies, 3GPP and associated vendors have been working on evolving cellular technologies to better address IoT requirements. The effort started with the definition of new LTE device categories. The aim was to both align with specific IoT requirements, such as low throughput and low power consumption, and decrease the complexity and cost of the LTE devices. This resulted in the definition of the LTE-M work item.

Because the new LTE-M device category was not sufficiently close to LPWA capabilities, in 2015 3GPP approved a proposal to standardize a new narrowband radio access technology called Narrowband IoT (NB-IoT). NB-IoT specifically addresses the requirements of a massive number of low-throughput devices, low device power consumption, improved indoor coverage, and optimized network architecture. The following sections review the proposed evolution of cellular technologies to better support the IoT opportunities by mobile service providers.

Standardization and Alliances

The 3GPP organization includes multiple working groups focused on many different aspects of telecommunications (for example, radio, core, terminal, and so on). Many service providers and vendors make up 3GPP, and the results of their collaborative work in these areas are the 3GPP specifications and studies. The workflow within 3GPP involves receiving contributions related to licensed LPWA work from the involved vendors. Then, depending on the access technology that is most closely aligned, such as 3G, LTE, or GSM, the IoT-related contribution is handled by either 3GPP or the GSM EDGE Radio Access Networks (GERAN) group.

Mobile vendors and service providers are not willing to lose leadership in this market of connecting IoT devices. Therefore, a couple intermediate steps have been pushed forward, leading to the final objectives set for NB-IoT and documented by 3GPP. At the same time, another industry group, the GSM Association (GSMA), has proposed the Mobile IoT Initiative, which “is designed to accelerate the commercial availability of LPWA solutions in licensed spectrum.” For more information on the Mobile IoT Initiative, go to www.gsma.com/connectedliving/mobile-iot-initiative/.

3.6.6.1 LTE Cat 0

The first enhancements to better support IoT devices in 3GPP occurred in LTE Release 12. A new user equipment (UE) category, Category 0, was added, with devices running at a maximum data rate of 1 Mbps. Generally, LTE enhancements target higher bandwidth improvements. Category 0 includes important characteristics to be supported by both the network and end devices. Meanwhile, the UE still can operate in existing LTE systems with bandwidths up to 20 MHz. These Cat 0 characteristics include the following:

- **Power saving mode (PSM):** This new device status minimizes energy consumption. Energy consumption is expected to be lower with PSM than with existing idle mode. PSM is defined as being similar to "powered off" mode, but the device stays registered with the network. By staying registered, the device avoids having to reattach or reestablish its network connection. The device negotiates with the network the idle time after which it will wake up. When it wakes up, it initiates a tracking area update (TAU), after which it stays available for a configured time and then switches back to sleep mode or PSM. A TAU is a procedure that an LTE device uses to let the network know its current tracking area, or the group of towers in the network from which it can be reached. Basically, with PSM, a device can be practically powered off but not lose its place in the network.
- **Half-duplex mode:** This mode reduces the cost and complexity of a device's implementation because a duplex filter is not needed. Most IoT endpoints are sensors that send low amounts of data that do not have a full-duplex communication requirement.

3.6.6.2 LTE-M

Following LTE Cat 0, the next step in making the licensed spectrum more supportive of IoT devices was the introduction of the LTE-M category for 3GPP LTE Release 13. These are the main characteristics of the LTE-M category in Release 13:

- **Lower receiver bandwidth:** Bandwidth has been lowered to 1.4 MHz versus the usual 20 MHz. This further simplifies the LTE endpoint.
- **Lower data rate:** Data is around 200 kbps for LTE-M, compared to 1 Mbps for Cat 0.
- **Half-duplex mode**: Just as with Cat 0, LTE-M offers a half-duplex mode that decreases node complexity and cost.
- **Image Enhanced discontinuous reception (eDRX):** This capability increases from seconds to minutes the amount of time an endpoint can "sleep" between paging cycles. A paging cycle is a periodic check-in with the network. This extended "sleep" time between paging cycles extends the battery lifetime for an endpoint significantly.

LTE-M requires new chipsets and additional software development. Commercial deployment is expected in 2017. Mobile carriers expect that only new LTE-M software will be required on the base stations, which will prevent re-investment in hardware.

3.6.6.3 NB-IoT

Recognizing that the definition of new LTE device categories was not sufficient to support LPWA IoT requirement, 3GPP specified Narrowband IoT (NB-IoT). The work on NB-IoT started with multiple proposals pushed by the involved vendors, including the following:

- Extended Coverage GSM (EC-GSM), Ericsson proposal
- Narrowband GSM (N-GSM), Nokia proposal
- Narrowband M2M (NB-M2M), Huawei/Neul proposal
- Narrowband OFDMA (orthogonal frequency-division multiple access), Qualcomm proposal
- Narrowband Cellular IoT (NB-CIoT), combined proposal of NB-M2M and NB-OFDMA
- Narrowband LTE (NB-LTE), Alcatel-Lucent, Ericsson, and Nokia proposal

- Cooperative Ultra Narrowband (C-UNB), Sigfox proposal

Consolidation occurred with the agreement to specify a single NB-IoT version based on orthogonal frequency-division multiple access (OFDMA) in the downlink and a couple options for the uplink. OFDMA is a modulation scheme in which individual users are assigned subsets of subcarrier frequencies. This enables multiple users to transmit low-speed data simultaneously. For more information on the uplink options, refer to the 3GPP specification TR 36.802.
Three modes of operation are applicable to NB-IoT:

- **Standalone**: A GSM carrier is used as an NB-IoT carrier, enabling reuse of 900 MHz or 1800 MHz.
- **In-band:** Part of an LTE carrier frequency band is allocated for use as an NB-IoT frequency. The service provider typically makes this allocation, and IoT devices are configured accordingly. Be aware that if these devices must be deployed across different countries or regions using a different service provider, problems may occur unless there is some coordination between the service providers, and the NB-IoT frequency band allocations are the same.
- **Guard band:** An NB-IoT carrier is between the LTE or WCDMA bands. This requires coexistence between LTE and NB-IoT bands.

In its Release 13, 3GPP completed the standardization of NB-IoT. Beyond the radio-specific aspects, this work specifies the adaptation of the IoT core to support specific IoT capabilities, including simplifying the LTE attach procedure so that a dedicated bearer channel is not required and transporting non-IP data. Subsequent releases of 3GPP NB-IoT will introduce additional features and functionality, such as multicasting, and will be backward compatible with Release 13.

Mobile service providers consider NB-IoT the target technology as it allows them to leverage their licensed spectrum to support LPWA use cases. For instance, NB-IoT is defined for a 200-kHz-wide channel in both uplink and downlink, allowing mobile service providers to optimize their spectrum, with a number of deployment options for GSM, WCDMA, and LTE spectrum, as shown in Figure 4-19.

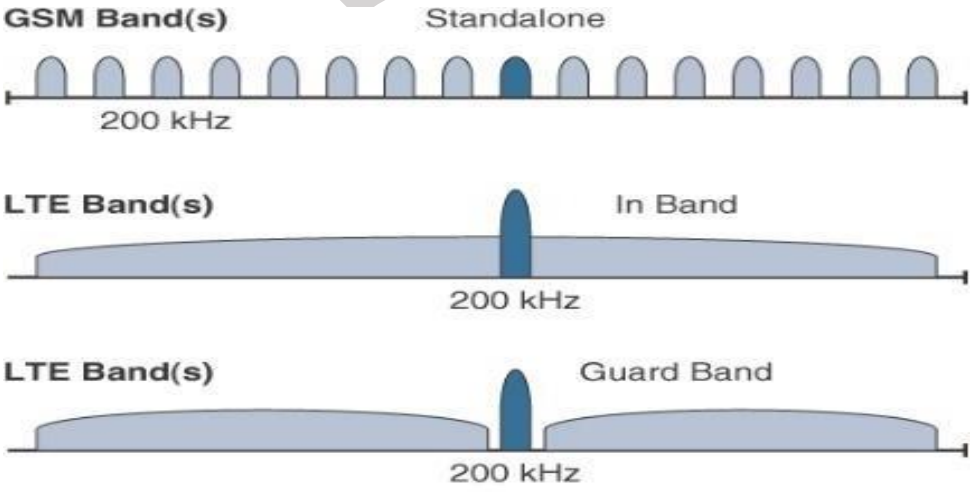


Figure 4-19 *NB-IoT Deployment Options*

In an LTE network, resource blocks are defined with an effective bandwidth of 180 kHz, while on NB-IoT, tone or subcarriers replace the LTE resource blocks. The uplink channel can be 15 kHz or
3.75 kHz or multi-tone (n*15 kHz, n up to 12). At Layer 1, the maximum transport block size (TBS) for downlink is 680 bits, while uplink is 1000 bits. At Layer 2, the maximum Packet Data Convergence Protocol (PDCP) service data unit (SDU) size is 1600 bytes. NB-IoT operates in half- duplex frequency-division duplexing (FDD) mode with a maximum data rate uplink of 60 kbps and downlink of 30 kbps.

Topology

NB-IoT is defined with a link budget of 164 dB; compare this with the GPRS link budget of 144 dB, used by many machine-to-machine services. The additional 20 dB link budget increase should guarantee better signal penetration in buildings and basements while achieving battery life requirements.

Competitive Technologies

In licensed bands, it is expected that 3GPP NB-IoT will be the adopted LPWA technology when it is fully available. Competitive technologies are mostly the unlicensed-band LPWA technologies such as LoRaWAN. The main challenge faced by providers of the licensed bands is the opportunity for non-mobile service providers to grab market share by offering IoT infrastructure without buying expensive spectrum.

MODULE 3

IP AS THE IOT NETWORKLAYER

Internet Protocol (IP), which has become the standard in many areas of IoT. With support from numerous standards and industry organizations, IP and its role as the network layer transport for IoT is a foundational element that has to be familiarized with.

3.1 The Business Case for IP

Data flowing from or to "things" is consumed, controlled, or monitored by data center servers either in the cloud or in locations that may be distributed or centralized. Dedicated applications are then run over virtualized or traditional operating systems or on network edge platforms (for example, fog computing). These lightweight applications communicate with thedata center servers. Therefore, thesystem solutions combining various physical and data link layers call for an architectural approach with a common layer(s) independent from the lower (connectivity) and/or upper (application) layers. This is how and why the Internet Protocol (IP) suite started playing a key architectural role in the early 1990s. IP was not only preferred in the IT markets but also for the OT environment.

3.1.1 The Key Advantages of Internet Protocol

One of themain differences between traditional information technology (IT) and operational technology (OT) is the lifetime of the underlying technologies and products. One way to guarantee multi-year lifetimes is to define a layered architecture such as the 30-year-old IP architecture. IP has largely demonstrated its ability to integrate small and large evolutions. At the same time, it is able to maintain its operations for large numbers of devices and users, such as the 3 billion Internet users. Before evaluating the pros and cons of IP adoption versus adaptation, this section provides a quick review of the key advantages of the IP suite for the Internet of Things:

- **Open and standards-based:** Operational technologies have often been delivered as turnkey features by vendors who may have optimized the communications through closed and proprietary networking solutions. The Internet of Things creates a new paradigm in which devices, applications, and users can leverage a large set of devices and functionalities while guaranteeing interchangeability and interoperability, security, and management. This calls for implementation, validation, and deployment of open, standards-based solutions. While manystandards development organizations (SDOs) are working on Internet of Things definitions, frameworks, applications, and technologies, none are questioning the role of the Internet Engineering Task Force (IETF) as the foundation for specifying and optimizing the network and transport layers. The IETF is an open standards body that focuses on the development of the Internet Protocol suite and related Internet technologies andprotocols.

- **Versatile:** A large spectrum of access technologies is available to offer connectivity of "things" in the last mile. Additional protocols and technologies are also used to transport IoTdata throughbackhaul links and in the data center. Even if physical and data link layers such as Ethernet, Wi-Fi, and cellular are widely adopted, the history of data communications demonstrates that no given wired or wireless

technology fits all deployment criteria. Furthermore, communication technologies evolve at a pace faster than the expected 10- to 20- year lifetime of OT solutions. So, the layered IP architecture is well equipped to cope with any type of physical and data link layers. This makes IP ideal as a long-term investment because various protocols at these layers can be used in a deployment now and over time, without requiring changes tothe whole solution architecture and data flow.

- **Ubiquitous:** All recent operating system releases, from general-purpose computers and servers to lightweight embedded systems (TinyOS, Contiki, and so on), have an integrated dual (IPv4 and IPv6) IP stack that gets enhanced over time. In addition, IoT application protocols in many industrial OT solutions have been updated in recent years to run over IP. While these updates have mostly consisted of IPv4 to this point, recent standardization efforts in several areas are adding IPv6.

- **Scalable:** As the common protocol of the Internet, IP has been massively deployed and tested for robust scalability. Millions of private and public IP infrastructure nodes have been operational for years, offering strong foundations for those not familiar with IP network management. Of course, adding huge numbers of "things" to private and public infrastructures may require optimizations and design rules specific to the newdevices.

- **Manageable and highly secure:** Communications infrastructure requires appropriate management and security capabilities for proper operations. One of the benefits that comes from 30 years of operational IP networks is the well-understood network management and security protocols, mechanisms, and toolsets that are widely available. Adopting IP network managementalso brings an operational business application to OT. Well-known network and security managementtools are easily leveraged with an IP network layer.

- **Stable and resilient:** IP has been around for 30 years, and it is clear that IP is a workable solution. IP has a large and well-established knowledge base and, more importantly, it has been used for years in critical infrastructures, such as financial and defense networks. In addition, IP has been deployed for critical services, such as voice and video, which have already transitioned from closed environments to open IP standards. Finally, its stability and resiliency benefit from the large ecosystem of IT professionals who can help design, deploy, and operate IP-based solutions.

- **Consumers' market adoption:** When developing IoT solutions and products targeting the consumer market, vendors knowthat consumers' access to applications and devices will occur predominantly over broadband and mobile wireless infrastructure. The main consumer devices range from smart phones to tablets and PCs. The common protocol that links IoT in the consumer space to these devices is IP.

- **The innovation factor:** The past two decades have largely established the adoption of IP as a factor for increased innovation. IP is the underlying protocol for applications ranging from file transfer and e-mail to the World Wide Web, e-commerce, social networking, mobility, and more. Even the recent computingevolution from PC to mobile and mainframes to cloud services are perfect demonstrations of the innovative ground enabled by IP.

3.1.2 Adoption or Adaptation of the Internet Protocol

How to implement IP in data center, cloud services, and operation centers hosting IoT applications mayseem

obvious, but the adoption of IP in the last mile is more complicated and often makes running IP end-to-end more difficult. The use of numerous network layer protocols in addition to IP is often a point of contention between computer networking experts. Typically, one of two models, adaptation or adoption, is proposed:

- ***Adaptation*** means application layered gateways (ALGs) must be implemented to ensure the translation between non-IP and IPlayers.
- ***Adoption*** involves replacing all non-IP layers with their IP layer counterparts, simplifying the deployment model and operations.

A similar transition is now occurring with IoT and its use of IP connectivity in the last mile. While IP is slowly becoming more prevalent, alternative protocol stacks are still often used. Let's look at a few examples in various industries tosee how IP adaptation and adoption are currentlyapplied to IoT last-mile connectivity.

In the industrial and manufacturing sector, there has been a move toward IP adoption. Solutions and product lifecycles in this space are spread over 10+ years, and many protocols have been developed for serial communications. While IP and Ethernet support were not specified in the initial versions, more recent specifications for these serial communications protocols integrate Ethernet and IPv4.

Supervisory control and data acquisition (SCADA) applications are typical examples of vertical market deployments that operate both the IP adaptation model and the adoption model. Found at the core of many modern industries, SCADA is an automation control system for remote monitoring and control of equipment. Implementations that make use of IP adaptation have SCADA devices attached through serial interfaces to a gateway tunneling or translating the traffic. With the IP adoption model, SCADA devices are attached via Ethernet to switches and routers forwarding their IPv4 traffic.

Another example is a ZigBee solution that runs a non-IP stack between devices and a ZigBee gateway that forwards traffic to an application server. A ZigBee gateway often acts as a translator between the ZigBee and IP protocol stacks. As highlighted by these examples, the IP adaptation versus adoption model still requires investigation for particular last-mile technologies used by IoT.

The following factors determine which model is best suited for last-mile connectivity:

- **Bidirectional versus unidirectional data flow:** While bidirectional communications are generally expected, some last-mile technologies offer optimization for unidirectional communication. For example, different classes of IoT devices, as defined in RFC 7228, may only infrequently need to report a few bytes of data to an application. These sorts of devices, particularly ones that communicate through LPWA technologies, include fire alarms sending alerts or daily test reports, electrical switches being pushed on or off, and water or gas meters sending weekly indexes. For these cases, it is not necessarily worth implementing a full IP stack. However, it requires the overall end-to-end architecture to solve potential drawbacks; for example, if there is only one-way communication to upload data to an application, then it is not possible to download new software or firmware to the devices. This makes integrating new features and bug and security fixes moredifficult.
- **Overhead for last-mile communications paths:** IP adoption implies a layered architecture with a per-packet overhead that varies depending on the IP version. IPv4 has 20 bytes of header at a minimum, and IPv6 has 40 bytes at the IP network layer. For the IP transport layer, UDP has 8 bytes of header overhead, while TCP has a minimum of 20 bytes. If the data to be forwarded by a device is

infrequent and only a few bytes, then it can potentially have more header overhead than device data—again, particularly in the case of LPWA technologies. Consequently, there is a need to decide whether the IP adoption model is necessary and, if it is, how it can be optimized. This same consideration applies to control plane traffic that is run over IP for low-bandwidth, last- mile links. Routing protocol and other verbose network services may either not be required or call for optimization.

- **Data flow model:** One benefit of the IP adoption model is the end-to-end nature of communications. Any node can easily exchange data with any other node in a network, although security, privacy, and other factors mayput controls and limits on the "end-to-end" concept. However, in many IoT solutions, a device's data flow is limited to one or two applications. In this case, theadaptation model can work because translation of traffic needs to occur only between the end device and one or two application servers. Depending on the network topology and the data flow needed, both IP adaptation and adoption models have roles to play in last- mile connectivity.
- **Network diversity:** One of the drawbacks of the adaptation model is a general dependency on single PHY and MAC layers. For example, ZigBee devices must only be deployed in ZigBee network islands. This same restriction holds for ITU G.9903 G3-PLC nodes. Therefore, a deployment must consider which applications have to run on the gateway connecting these islands and the rest of the world. Integration and coexistence of new physical and MAC layers or new applications impact how deployment and operations have to be planned. This is not a relevant consideration for the adoption model.

3.2 The Need for Optimization

The following sections take a detailed look at why optimization is necessary for IP. Both the nodes and the network itself can often be constrained in IoT solutions. Also, IP is transitioning from version 4 to version 6, which can add further confinements in the IoT space.

3.2.1 Constrained Nodes

Another limit is that this network protocol stack on an IoT node may be required to communicate through an unreliable path. Even if a full IP stack is available on the node, this causes problems such as limited or unpredictable throughput and low convergence when a topology change occurs.

Finally, power consumption is a key characteristic of constrained nodes. Many IoT devices are battery powered, with lifetime battery requirements varying from a few months to 10+ years. This drives theselection of networking technologies since high-speed ones, such as Ethernet, Wi-Fi, and cellular, are not (yet) capable of multi-year battery life. Current capabilities practically allow less than a year for these technologies on battery-powered nodes. Of course, power consumption is muchless of a concern on nodes that do not require batteries as an energy source.

The power consumption requirements on battery-powered nodes impact communication intervals. To help extend battery life, enable a "low-power" mode instead of one that is "always on." Another option is "always off," which means communications are enabled only when needed to senddata.

While it has been largely demonstrated that production IP stacks perform well in constrained nodes,

classification of these nodes helps when evaluating the IP adoption versus adaptation model selection. IoT constrained nodes can be classified as follows:

- **Devices that are very constrained in resources, may communicate infrequently to transmit a few bytes, and may have limited security and management capabilities:** This drives the need for the IP adaptation model, where nodes communicate through gateways and proxies.
- **Devices with enough power and capacities to implement a stripped-down IP stack or non- IP stack:** In this case, either an optimized IP stack and directly communicate with application servers (adoption model) or go for an IP or non-IP stack and communicate through gateways and proxies (adaptation model) can be implemented.
- **Devices that are similar to generic PCs in terms of computing and power resources but have constrained networking capacities, such as bandwidth:** These nodes usually implement a full IP stack (adoption model), but network design and application behaviors must cope with the bandwidth constraints.

3.2.2 Constrained Networks

Constrained networks have unique characteristics and requirements. In contrast with typical IP networks, where highly stable and fast links are available, constrained networks are limited by low- power, low-bandwidth links (wireless and wired). They operate between a few kbps and a few hundred kbps and may utilizea star, mesh, or combinednetwork topologies, ensuringproper operations. Witha constrained network,in addition to limited bandwidth, itis not unusualfor thepacket delivery rate (PDR) to oscillate between low and high percentages. Large bursts of unpredictable errors and even loss of connectivity at times may occur. These behaviors can be observed on both wireless and narrowband power-line communication links, where packet delivery variation may fluctuate greatly during the course of aday.

Unstable link layer environments create other challenges in terms of latency and control plane reactivity. One of the golden rules in a constrained network is to "underreact to failure." Due to the low bandwidth, a constrained network that overreacts can lead to a network collapse—which makes the existing problem worse. Control plane traffic must also be kept at a minimum; otherwise, it consumes the bandwidth that is needed by the data traffic. Finally, the power consumption in battery-powered nodes has to be considered. Any failure or verbose control plane protocol may reduce the lifetime of the batteries.

3.2.3 IP Versions

For 20+ years, the IETF has been working on transitioning the Internet from IP version 4 to IP version 6. The main driving force has been the lack of address space in IPv4 as the Internet has grown. IPv6 has a much larger range of addresses that should not be exhausted for the foreseeable future. Today, both versions of IP run over the Internet, but most traffic is still IPv4 based.

While it may seem natural to base all IoT deployments on IPv6, current infrastructures and their associated lifecycle of solutions, protocols, and products need to be taken into account. IPv4 is entrenched in these current infrastructures, and so support for it is required in most cases. Therefore, the Internet of Things

has to follow a similar path as the Internet itself and support both IPv4 and IPv6 versions concurrently. Techniques such as tunneling and translation need to be employed in IoT solutions to ensure interoperability between IPv4 and IPv6.

A variety of factors dictate whether IPv4, IPv6, or both can be used in an IoT solution. Most often these factors include a legacy protocol or technology that supports only IPv4. Newer technologies and protocols almost always support both IP versions.

The following are some of the main factors applicable to IPv4 and IPv6 support in an IoT solution:

- **Application Protocol:** IoT devices implementing Ethernet or Wi-Fi interfaces can communicate over both IPv4 and IPv6, but the application protocol may dictate the choice of the IP version. For IoT devices with application protocols defined by the IETF, such as HTTP/HTTPS, CoAP, MQTT, and XMPP, both IP versions are supported. The selection of the IP version is only dependent on the implementation.
- **Cellular Provider and Technology:** IoT devices with cellular modems are dependent on the generation of the cellular technology as well as the data services offered by the provider. For the first three generations of data services—GPRS, Edge, and 3G—IPv4 is the base protocol version. Consequently, if IPv6 is used with these generations, it must be tunneled over IPv4. On 4G/LTE networks, data services can use IPv4 or IPv6 as a base protocol, depending on the provider.
- **Serial Communications:** Many legacy devices in certain industries, such as manufacturing and utilities, communicate through serial lines. Data is transferred using either proprietary or standards-based protocols, such as DNP3, Modbus, or IEC 60870-5-101. In the past, communicating this serial data over anysort of distance could be handled by an analog modem connection. However, as service provider support for analog line services has declined, the solution for communicating with these legacy devices has been to use local connections. To make this work, connect the serial port of the legacy device to a nearby serial port on a piece of communications equipment, typically a router. This local router then forwards the serial traffic over IP to the central server for processing. Encapsulation of serial protocols over IP leverages mechanisms such as raw socket TCP or UDP. While raw socket sessions can run over both IPv4 and IPv6, current implementations are mostly available for IPv4 only.
- **IPv6 Adaptation Layer:** IPv6-only adaptation layers for some physical and data link layers for recently standardized IoT protocols support only IPv6. While themost commonphysical and data link layers (Ethernet, Wi-Fi, and so on) stipulate adaptation layers for both versions, newer technologies, such as IEEE 802.15.4 (Wireless Personal Area Network), IEEE 1901.2, and ITU G.9903 (Narrowband Power Line Communications) only have an IPv6 adaptation layer specified. This means that any device implementing a technology that requires an IPv6 adaptation layer must communicate over an IPv6-only subnetwork. This is reinforced by the IETF routing protocol for LLNs, RPL, which is IPv6 only.

3.3 Optimizing IP for IoT

While the Internet Protocol is key for a successful Internet of Things, constrained nodes and constrained networks mandate optimization at various layers and on multiple protocols of the IP architecture. The following sections introduce some of these optimizations already available from the market or under development by the IETF. Figure 3.1 highlights the TCP/IP layers where optimization is applied.

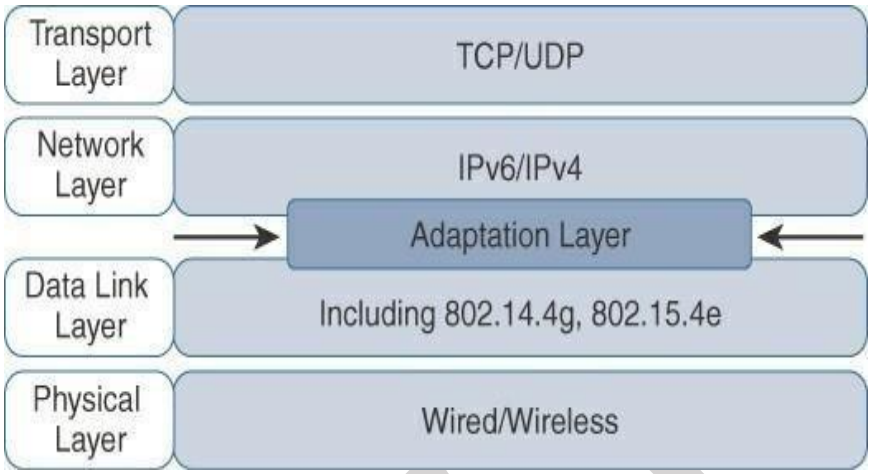


Figure 3.1 *Optimizing IP for IoT Using an Adaptation Layer*

3.3.1 From 6LoWPAN to 6Lo

In the IP architecture, the transport of IP packets over any given Layer 1 (PHY) and Layer 2 (MAC) protocol must be defined and documented. The model for packaging IP into lower-layer protocols is often referred to as an *adaptation layer*.

Unless the technology is proprietary, IP adaptation layers are typically defined by an IETF working group and released as a Request for Comments (RFC). An RFC is a publication from the IETF that officially documents Internet standards, specifications, protocols, procedures, and events. For example, RFC 864 describes how an IPv4 packet gets encapsulated over an Ethernet frame, and RFC 2464 describes how the same function is performed for an IPv6 packet.

IoT-related protocols follow a similar process. The main difference is that an adaptation layer designed for IoT may include some optimizations to deal with constrained nodes and networks. The main examples of adaptation layers optimized for constrained nodes or "things" are the ones under the 6LoWPANworking group and its successor, the 6Lo working group. The initial focus of the 6LoWPAN working group was to optimize the transmission of IPv6 packets over constrained networks such as IEEE 802.15.4. Figure 3.2 shows an example of an IoT protocol stack using the 6LoWPAN adaptation layer beside the well- known IP protocol stack for reference.

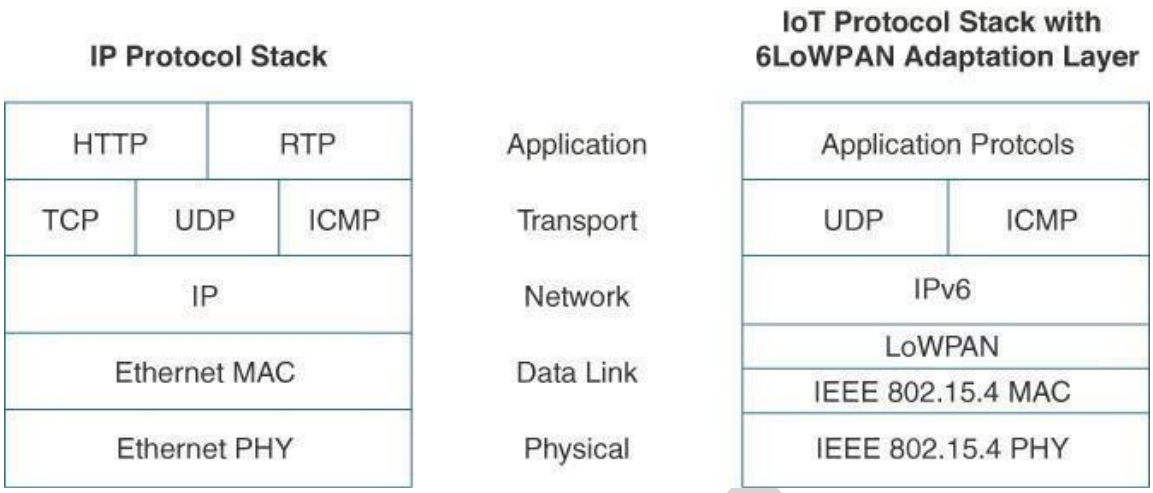


Figure 3.2 *Comparison of an IoT Protocol Stack Utilizing 6LoWPAN and an IP Protocol Stack*

The 6LoWPAN working group published several RFCs, but RFC 4994 is foundational because it defines frame headers for the capabilities of header compression, fragmentation, and mesh addressing. These headers can be stacked in the adaptation layer to keep these concepts separate while enforcing a structured method for expressing each capability. Depending on the implementation, all, none, or any combination of these capabilities and their corresponding headers can beenabled.
Figure 3.3 shows some examples of typical 6LoWPAN header stacks.

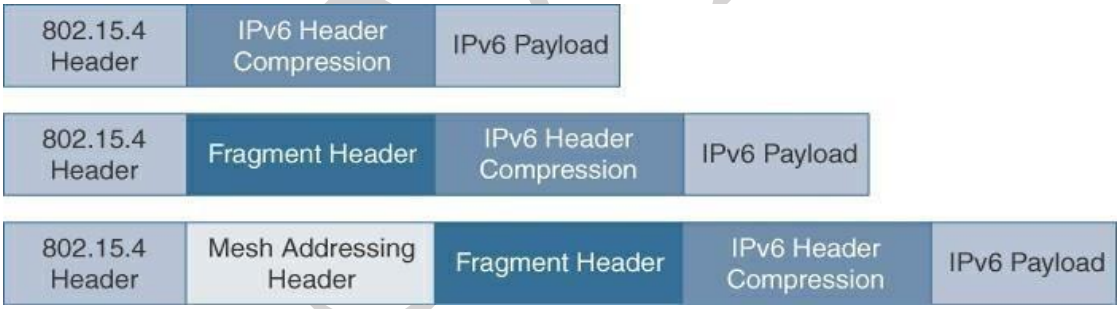


Figure 3.3 *6LoWPAN Header Stacks*

3.3.1.1 Header Compression

IPv6 header compression for 6LoWPANwas defined initially in RFC 4944 and subsequentlyupdated by RFC 6282. This capability shrinks the size of IPv6's 40-byte headers and User Datagram Protocol's (UDP's) 8-byte headers down as low as 6 bytes combined in some cases.

6LoWPAN header compression is stateless, and conceptually it is not too complicated. However, a number of factors affect the amount of compression, such as implementation of RFC 4944 versus RFC 6922, whether UDP is included, and various IPv6 addressing scenarios. It is beyond the scope of this book to cover every use case and how the header fields change for each. At a high level, 6LoWPAN works by taking advantage of shared information known by all nodes from their participation in the local network. In addition, it omits some standard header fields by assuming commonly used values. Figure 3.4 highlights an example that shows the amount of reduction that is possible with 6LoWPAN header compression.

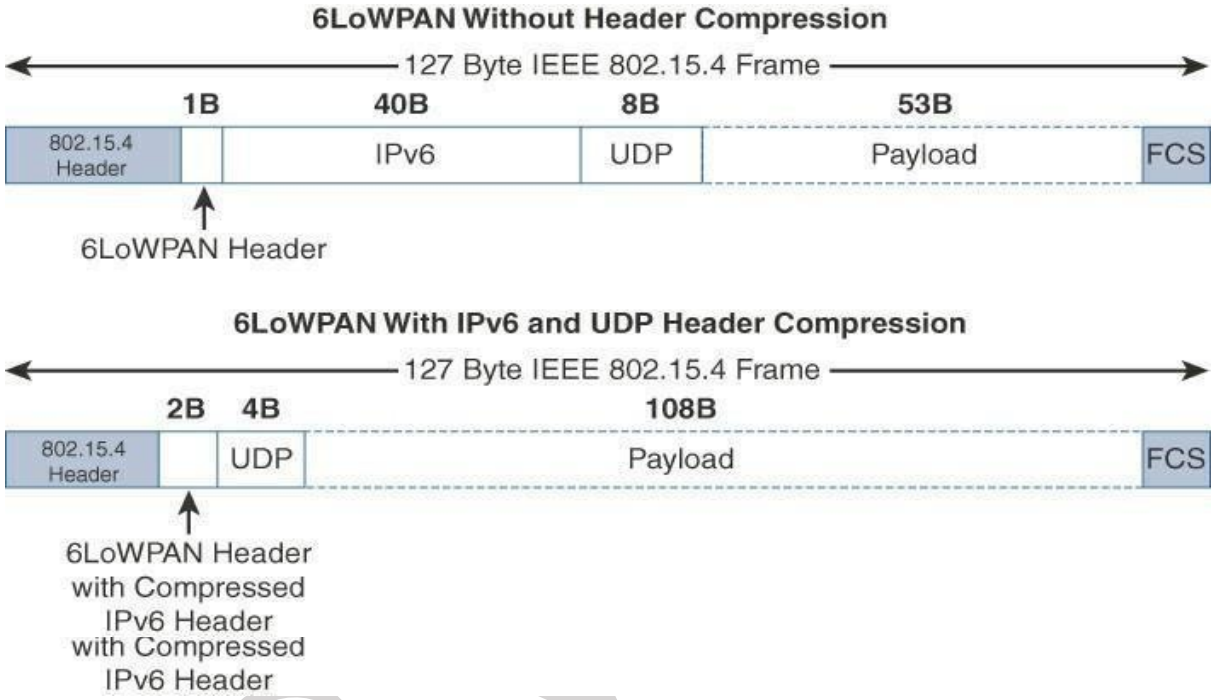


Figure 3.4 *6LoWPAN Header Compression*

At the top of Figure 3.4, a 6LoWPAN frame without any header compression enabled: The full 40-byte IPv6 header and 8-byte UDP header are visible. The 6LoWPAN header is only a single byte in this case. Notice that uncompressed IPv6 and UDP headers leave only 53 bytes of data payload out of the 127-byte maximum frame size in the case of IEEE 802.15.4.

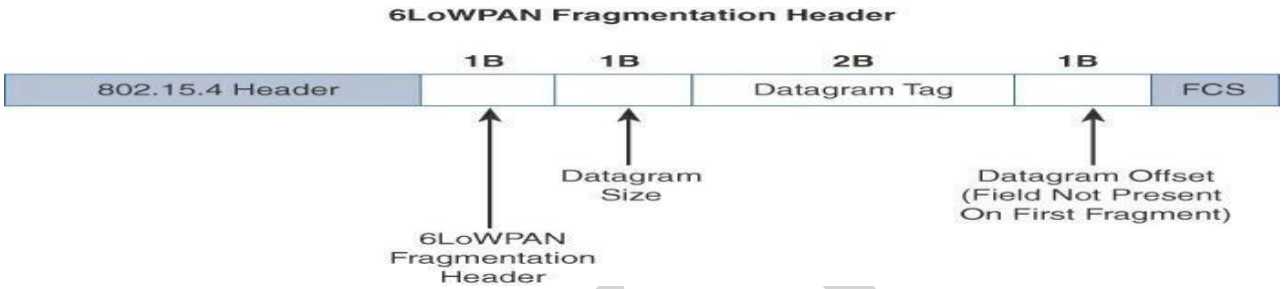
The bottom half of Figure 3.4 shows a frame where header compression has beenenabled for a best- case scenario. The 6LoWPAN header increases to 2 bytes to accommodate the compressed IPv6 header, and UDP has been reduced in half, to 4 bytes from 8. Most importantly, theheader compression has allowed thepayload to more than double, from 53 bytes to 108 bytes, which is obviously muchmore efficient. Note that the 2-byte header compression applies to intra-cell communications, while communications external to the cell may require some field of the header to not be compressed.

3.3.1.2 Fragmentation

The maximum transmission unit (MTU) for an IPv6 network must be at least 1280 bytes. The term *MTU*

defines the size of the largest protocol data unit that can be passed. For IEEE 802.15.4, 127 bytes is the MTU. Because of this is there is a problem that IPv6 with a much larger MTU, is carried inside the 802.15.4 frame with a much smaller one. To remedy this situation, large IPv6 packets must be fragmented across multiple 802.15.4 frames at Layer 2.

The fragment header utilized by 6LoWPAN is composed of three primary fields: Datagram Size, Datagram Tag, and Datagram Offset. The 1-byte Datagram Size field specifies the total size of theunfragmented payload. Datagram Tag identifies the set of fragments for a payload. Finally, the Datagram Offset field delineates how far into a payload a particular fragment occurs. Figure 3.5 provides an overview of a



6LoWPAN fragmentation header.

Figure 3.5 *6LoWPAN Fragmentation Header*

In Figure 3.5, the 6LoWPAN fragmentation header field itself uses a unique bit value to identify that the subsequent fields behind it are fragment fields as opposed to another capability, such as header compression. Also, in the first fragment, the Datagram Offset field is not present because it would simply be set to 0. This results in the first fragmentation header for an IPv6 payload being only 4 bytes long. The remainder of the fragments has a 5-byte header field so that the appropriate offset can be specified.

3.3.1.3 Mesh Addressing

The purpose of the 6LoWPANmesh addressing function is to forward packets over multiple hops. Three fields are defined for this header: Hop Limit, Source Address, and Destination Address. Analogous to the IPv6 hop limit field, the hop limit for mesh addressing also provides an upper limit on how many times the frame can be forwarded. Each hop decrements this value by 1 as it is forwarded. Once the value hits 0, it is dropped and no longer forwarded. The Source Address and Destination Address fields for mesh addressing are IEEE 802.15.4 addresses indicating the endpoints of an IP hop. Figure 3.6 details the 6LoWPAN mesh addressing header fields.

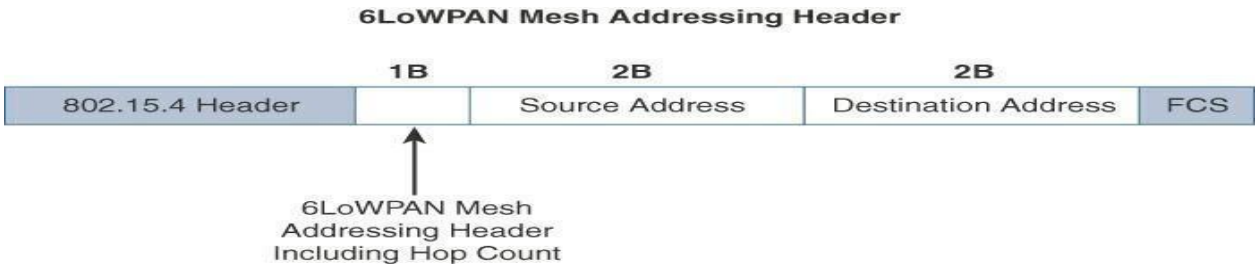


Figure 3.6 *6LoWPAN Mesh Addressing Header*

Mesh-Under Versus Mesh-Over Routing

For network technologies such as IEEE 802.15.4, IEEE 802.15.4g, and IEEE 1901.2a that support mesh topologies and operate at the physical and data link layers, two main options exist for establishing reachability and forwarding packets. With the first option, mesh-under, the routing of packets is handled at the 6LoWPAN adaptation layer. The other option, known as "**mesh-over**" or "route-over," utilizes IP routing for getting packets to their destination.

The term ***mesh-under*** is used because multiple link layer hops can be used to complete a single IP hop. Nodes have a Layer 2 forwarding table that they consult to route the packets to their final destination within the mesh. An edge gateway terminates the mesh-under domain. The edge gateway must also implement a mechanism to translate between the configured Layer 2 protocol and any IP routing mechanism implemented on other Layer 3 IP interfaces.

In mesh-over or route-over scenarios, IP Layer 33 routing is utilized for computing reachability and then getting packets forwarded to their destination, either inside or outside the mesh domain. Each full-functioning node acts as an IP router, so each link layer hop is an IP hop. When a LoWPAN has been implemented using different link layer technologies, a mesh-over routing setup is useful. While traditional IP routing protocols can be used, a specialized routing protocol for smart objects, such as RPL, is recommended.

3.3.2 6Lo Working Group

With the work of the 6LoWPAN working group completed, the 6Lo working group seeks to expand on this completed work with a focus on IPv6 connectivity over constrained-node networks. While the 6LoWPAN working group initially focused its optimizations on IEEE 802.15.4 LLNs, standardizing IPv6 over other link layer technologies is still needed.

Therefore, the charter of the 6Loworking group, now called the IPv6 over Networks of Resource- Constrained Nodes, is to facilitate the IPv6 connectivityover constrained-node networks. In particular, this working group is focused on the following:

- **IPv6-over-foo adaptation layer specifications using 6LoWPAN technologies (RFC4944, RFC6282, RFC6775) for link layer technologies:** For example, this includes:
    - IPv6 over Bluetooth Low Energy
    - Transmission of IPv6 packets over near-field communication IPv6 over 802.11ah

- Transmission of IPv6 packets over DECT Ultra Low Energy
- Transmission of IPv6 packets on WIA-PA (Wireless Networks for Industrial Automation– Process Automation)
- Transmission of IPv6 over Master Slave/Token Passing (MS/TP)

- **Information and data models such as MIB modules:** One example is RFC 7388, "Definition of Managed Objects for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)."
- **Optimizations that are applicable to more than one adaptation layer specification:** For example, this includes RFC 7400, "6LoWPAN-GHC: Generic Header Compression for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)."
- **Informational and maintenance publications needed for the IETF specifications in this area**

3.3.3 6TiSCH

Manyproprietary wireless technologies have been developed and deployed in various industry verticals over the years. However, the publication of the IEEE 802.15.4 physical and data link layer specifications, followed by IEEE 802.15.4e amendments, has opened thepath to standardized, deterministic communications over wireless networks. IEEE 802.15.4e, Time-Slotted Channel Hopping (TSCH), is an add-on to the Media Access Control (MAC) portion of the IEEE 802.15.4 standard, with direct inheritance from other standards, such as WirelessHART and ISA100.11a.

Devices implementing IEEE 802.15.4e TSCH communicate by following a Time Division Multiple Access (TDMA) schedule. An allocation of a unit of bandwidth or time slot is scheduled between neighbor nodes. This allows the programming of predictable transmissions and enables deterministic, industrial-type applications. In comparison, other 802.15.4 implementations do not allocate slices of bandwidth, so communication, especially during times of contention, maybe delayed or lost because it is always best effort.

To standardize IPv6 over the TSCH mode of IEEE 802.15.4e (known as 6TiSCH), the IETF formed the 6TiSCH working group. This working group works on the architecture, information model, and minimal 6TiSCH configuration, leveraging and enhancing work done by the 6LoWPAN working group, RoLL working group, and CoRE working group. The RoLL working group focuses on Layer 3 routing for constrained networks.

An important element specified by the6TiSCH working group is 6top, a sublayer that glues together the MAC layer and 6LoWPANadaptation layer. This sublayer provides commands to the upper network layers, such as RPL. In return, these commands enable functionalities including network layer routing decisions, configuration, and control procedures for 6TiSCH schedule management.

The IEEE 802.15.4e standard defines a time slot structure, but it does not mandate a scheduling algorithm for how the time slots are utilized. This is left to higher-level protocols like 6TiSCH. Scheduling is critical because it can affect throughput, latency, and power consumption. Figure 3.7 shows where 6top resides in relation to IEEE 802.15.4e, 6LoWPAN HC, and IPv6. HC (Header Compression)

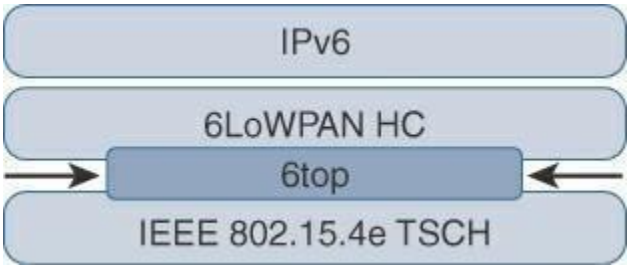


Figure 5-7 *Location of 6TiSCH's 6top Sublayer*

Schedules in 6TiSCH are broken down into cells. A cell is simply a single element in the TSCH schedule that can be allocated for unidirectional or bidirectional communications between specific nodes. Nodes only transmit when the schedule dictates that their cell is open for communication. The 6TiSCH architecture defines four schedule managementmechanisms:

- **Static scheduling:** All nodes in theconstrained network share a fixed schedule. Cells are shared, and nodes contend for slot access in a slotted aloha manner. Slotted aloha is a basic protocol for sending data using time slot boundaries when communicating over a shared medium. Static scheduling is a simple scheduling mechanism that can be used upon initial implementation or as a fallback in the case of network malfunction. The drawback with static scheduling is that nodes mayexpect a packet at any cell in the schedule. Therefore, energy is wasted idly listening across all cells.
- **Neighbor-to-neighbor scheduling:** A schedule is established that correlates with the observed number of transmissions between nodes. Cells in this schedule can be added or deleted as traffic requirements and bandwidth needs change.
- **Remote monitoring and scheduling management:** Time slots and other resource allocation are handled by a management entity that can be multiple hops away. The scheduling mechanism leverages 6top and even CoAP in some scenarios. This scheduling mechanism provides quite a bit of flexibility and control in allocating cells for communication betweennodes.
- **Hop-by-hop scheduling:** A node reserves a path to a destination node multiple hops away by requesting the allocation of cells in a schedule at each intermediate node hop in thepath. The protocol that is used by a node to trigger this scheduling mechanism is not defined at this point.

In addition to schedule management functions, the 6TiSCH architecture also defines three different forwarding models. Forwarding is the operation performed on each packet by a node that allows it to be delivered to a next hop or an upper-layer protocol. The forwarding decision is based on a preexisting state that was learned from a routing computation. There are three 6TiSCH forwarding models:

- **Track Forwarding (TF):** This is the simplest and fastest forwarding model. A "track" in this modelis a unidirectional path between a source and a destination. This track is constructed by pairing bundles of receive cells in a schedule with a bundle of receive cells set to transmit. So, a frame received within a particular cell or cell bundle is switched to another cell or cell bundle. This forwarding occurs regardless of the network layer protocol.

- **Fragment forwarding (FF):** This model takes advantage of 6LoWPAN fragmentation to build a Layer 2 forwarding table. IPv6 packets can get fragmented at the 6LoWPAN sublayer to handle the differences between IEEE 802.15.4 payload size and IPv6 MTU. Additional headers for RPL source route information can further contribute to the need for fragmentation. However, with FF, a mechanism is defined where the first fragment is routed based on the IPv6 header present. The 6LoWPANsublayer learns the next-hop selection of this first fragment, which is then applied to all subsequent fragments of that packet. Otherwise, IPv6 packets undergo hop-by-hop reassembly. This increases latency and can be power- and CPU-intensive for a constrained node.
- **IPv6 Forwarding(6F):** This model forwards traffic based on its IPv6 routing table. Flows of packets should be prioritized by traditional QoS (quality of service) and RED (random early detection) operations. QoS is a classification scheme for flows based on their priority, and RED is a common congestion avoidance mechanism.

For many IoT wireless networks, it is not necessary to be able to control the latency and throughput for sensor data. However, when some sort of determinism is needed, 6TiSCH provides an open, IPv6-based standard solution for ensuring predictable communications over wireless sensor networks. However, its adoption by the industry is still an ongoing effort.

3.3.4 RPL

The IETF chartered the RoLL (Routing over Low-Power and Lossy Networks) working group to evaluate all Layer 3 IP routing protocols and determine the needs and requirements for developing a routing solution for IP smart objects. After study of various use cases and a survey of existing protocols, the consensus was that a new routing protocol should be developed for use by IP smart objects, given the characteristics and requirements of constrained networks. This new distance-vector routing protocol was named the **IPv6 Routing Protocol for Low Power and LossyNetworks (RPL).**The RPL specification was published as RFC 6550 by the RoLL working group. In an RPL network, each node acts as a router and becomes part of a mesh network. Routing is performed at the IP layer. Each node examines every received IPv6 packet and determines the next- hop destination based on the information contained in the IPv6 header. No information fromthe MAC-layer header is needed to perform next-hop determination. To cope with the constraints of computing and memory that are common characteristics of constrained nodes, the protocol defines two modes:

- **Storing mode:** All nodes contain the full routing table of the RPL domain. Every node knows how to directly reach every other node.
- **Non-storing mode:** Only the border router(s) of the RPL domain contain(s) the full routing table. All other nodes in the domain only maintain their list of parents and use this as a list of default routes toward the border router. This abbreviated routing table saves memory space and CPU. When communicating in non-storing mode, a node always forwards its packets to the border router, which knows how to ultimately reach the finaldestination.

RPLis based on the concept of a directed acyclic graph (DAG). A DAG is a directed graph where no cycles exist. This means that from any vertex or point in the graph, it cannot follow an edge or a line back to this same point. All of the edges are arranged in paths oriented toward and terminating at one or more root nodes. Figure 3.8 shows a basic DAG.

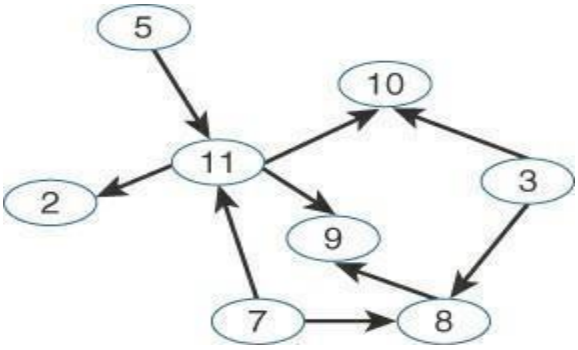


Figure 3.8 *Example of a Directed Acyclic Graph (DAG)*

A basic RPL process involves building a destination-oriented directed acyclic graph (DODAG). A DODAG is a DAG rooted to one destination. In RPL, this destination occurs at a border router known as the DODAG root. Figure 3.9 compares a DAG and a DODAG. From figure a DAG has multiple roots, whereas the DODAG has just one.

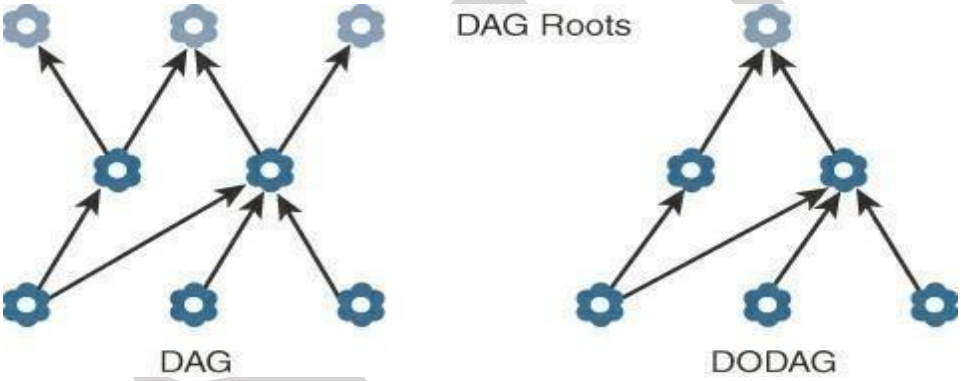


Figure 5-9 *DAG and DODAG Comparison*

- In a DODAG, each node maintains up to three parents that provide a path to the root. Typically, one of these parents is thepreferred parent, which means it is thepreferred next hop for upward routes toward the root.
- The routing graph created by theset of DODAG parents across all nodes defines the full set of upward routes. RPL protocol implementation should ensure that routes are loop free by disallowing nodes from selected DODAG parents that are positioned further away from the border router.
- Upward routes in RPL are discovered and configured using DAG Information Object (DIO) messages. Nodes listen to DIOs to handle changes in the topology that can affect routing. The information in DIO messages determines parents and the best path to the DODAG root.
- Nodes establish downward routes by advertising their parent set toward the DODAG root using a Destination Advertisement Object (DAO) message. DAO messages allow nodes to inform their parents

of their presence and reachability to descendants.

- In the case of the non-storing mode of RPL, nodes sending DAO messages report their parent sets directly to the DODAG root (border router), and only the root stores the routing information. The root uses the information to then determine source routes needed for delivering IPv6 datagrams to individual nodes downstream in the mesh.
- For storing mode, each node keeps track of the routing information that is advertised in the DAO messages. While this is more power- and CPU-intensive for each node, the benefit is that packets can take shorter paths between destinations in the mesh. The nodes can make their own routing decisions; in non-storing mode, on the other hand, all packets must go up to the root to get a route for moving downstream.
- RPL messages, such as DIO and DAO, run on top of IPv6. These messages exchange and advertise downstream and upstream routing information between a border router and the nodes under it. As illustrated in Figure 3.10, DAO and DIO messages move both up and down the DODAG, depending on the exact message type.

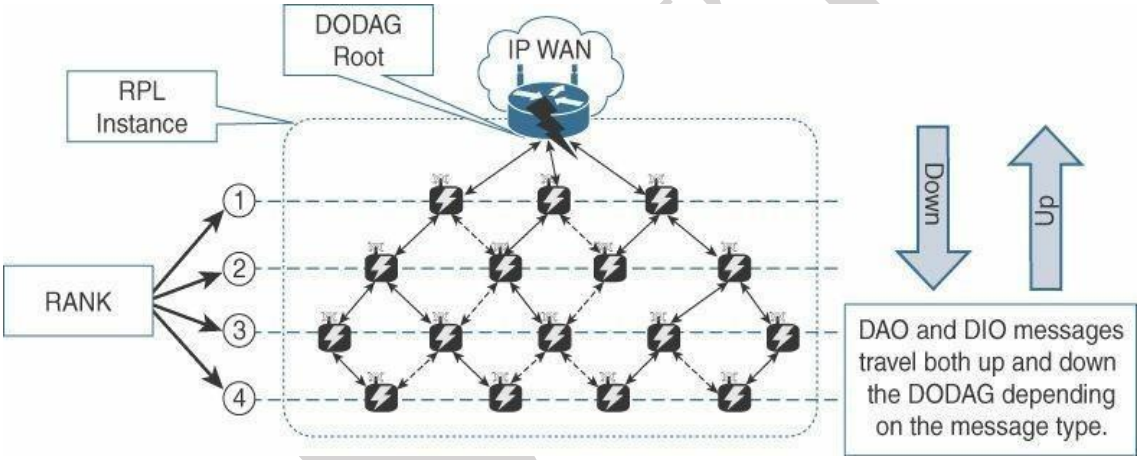


Figure 3.10 *RPL Overview*

Objective Function (OF)

An objective function(OF) defines how metrics are used to select routes and establish a node's rank. Standards such as RFC 6552 and 6719 have been published to document OFs specific to certain use cases and node types.

For example, nodes implementing an OF based on RFC 6719's Minimum Expected Number of Transmissions (METX) advertise the METX among their parents in DIO messages. Whenever a node establishes its rank, it simply sets the rank to the current minimum METX among its parents.

Rank

The rank is a rough approximation of how "close" a node is to the root and helps avoid routing loops and the count-to-infinity problem. Nodes can only increase their rank when receiving a DIO message with a larger version number. However, nodes maydecrease their rank whenever they have established lower-cost routes. While the rank and routing metrics are closely related, the rank differs from routing metrics in that it is used

as a constraint to prevent routing loops.

RPL Headers

Specific network layer headers are defined for datagrams being forwarded within an RPL domain. One of the headers is standardized in RFC 6553, "The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams," and the other is discussed in RFC 6554, "An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL)."

RFC 6553 defines a new IPv6 option, known as the RPL option. The RPL option is carried in the IPv6 Hop-by-Hop header. The purpose of this header is to leverage data-plane packets for loop detection in a RPL instance. As discussed earlier, DODAGs only have single paths and should be loop free.

RFC 6554 specifies the Source Routing Header (SRH) for use between RPL routers. Aborder router or DODAG root inserts the SRH when specifying a source route to deliver datagrams to nodes downstream in the mesh network.

Metrics

RPL defines a large and flexible set of new metrics and constraints for routing in RFC 6551. Developed to support powered and battery-powered nodes, RPL offers a far more complete set than any other routing protocol. Some of the RPL routing metrics and constraints defined in RFC 6551 include the following:

- **Expected Transmission Count(ETX):** Assigns a discrete value to the number of transmissions a node expects to make to deliver apacket.
- **Hop Count:** Tracks the number of nodes traversed in a path. Typically, a path with a lower hop count is chosen over a path with a higher hop count.
- **Latency:** Varies depending on power conservation. Paths with a lower latency are preferred.
- **Link Quality Level:** Measures the reliability of a link by taking into account packet error rates caused by factors such as signal attenuation and interference.
- **Link Color:** Allows manual influence of routing by administratively setting values to make a link more or less desirable. These values can be either statically or dynamically adjusted for specific traffic types.
- **Node State and Attribute:** Identifies nodes that function as traffic aggregators and nodes that are being impacted byhigh workloads. High workloads could be indicative of nodes that have incurred high CPU or low memory states. Naturally, nodes that are aggregators are preferred over nodes experiencing high workloads.
- **Node Energy:** Avoids nodes with low power, so a battery-powered node that is running out of energy can be avoided and the life of that node and the network can be prolonged.
- **Throughput:** Provides the amount of throughput for a node link. Often, nodes conserving power use lower throughput. This metric allows the prioritization of paths with higher throughput.

In addition to the metrics and constraints listed in RFC 6551, others can also be implemented. For example, let's look at a scenario in which two constraints are used as a filter for pruning links that do not satisfy the specified conditions. One of the constraints is ETX. ETX, which is described in RFC 6551. The other constraint, Relative Signal Strength Indicator (RSSI), specifies the power present in a received radio signal. Signals with low strength are generally less reliable and more susceptible to interference, resulting in packet loss.

In this scenario, a DODAG root and nodes form an IEEE 802.15.4 mesh. When a node finds a potential parent, it enters the neighbor into its routing table. However, it does not yet use the new neighbor for routing. Instead, the node mustfirst establish that thelink quality to its neighbor is sufficient for forwarding datagrams. The node determines whether the link quality to a potential parent is sufficient by looking at its programmed constraints. In this example, the configured constraints are ETX and RSSI. If the RSSI in both directions exceeds a threshold and the ETX falls below a threshold, then the node confirms that the link quality to the potential parent is sufficient.

Once a node has determined that the link quality to a potential parent is sufficient, it adds the appropriate default route entry to its forwarding table. Maintaining RSSI and ETX for neighboring nodes is done at the link layer and stored in the link layer neighbor table. The results from all link layer unicast traffic are fed into the RSSI and ETX computation for neighboring devices. If the link quality is not sufficient, then the link is not addedto the forwarding table and is therefore not used for routing packets.Authentication and Encryption on Constrained Nodes

IoT security is a complex topic that often spawns discussions and debates across the industry. So it is worth mentioning here the IETF workinggroups that are focused on their security: ACE and DICE.

ACE

Much like the RoLL working group, the Authentication and **Authorization for Constrained Environments (ACE)** working group is tasked with evaluating the applicability of existing authentication and authorization protocols and documenting their suitability for certain constrained- environment use cases. Once thecandidate solutions are validated, the ACE working group will focus its work on CoAP with the Datagram Transport Layer Security (DTLS) protocol. The ACE working group mayinvestigate other security protocols later, with a particular focus on adapting whatever solution is chosen to HTTP and TLS.

The ACE working group expects to produce a standardized solution for authentication and authorization that enables authorized access (Get, Put, Post, Delete) to resources identified by a URI and hosted on a resource server in constrained environments. An unconstrained authorization server performs mediation of the access. Aligned with the initial focus, access to resources at a resource server by a client device occurs using CoAP and is protected byDTLS.

DICE

New generations of constrained nodes implementing an IP stack over constrained access networks are expected to run an optimized IP protocol stack. For example, when implementing UDP at the transport layer, the IETF Constrained Application Protocol (CoAP) should be used at the application layer.

In constrained environments secured by DTLS, CoAP can be used to control resources on a device.

The **DTLS in Constrained Environments (DICE)** working group focuses on implementing the DTLS

transport layer security protocol in these environments. The first task of the DICE working group is to define an optimized DTLS profile for constrained nodes. In addition, the DICE working group is considering the applicability of the DTLS record layer to secure multicast messages and investigating how the DTLS handshake in constrained environments can get optimized.

3.4 Profiles and Compliances

Leveraging the Internet Protocol suite for smart objects involves a collection of protocols and options that must work incoordination with lower and upper layers.
Therefore, profile definitions, certifications, and promotion by alliances can help implementers develop solutions that guarantee interoperability and/or interchangeability of devices.

This section introduces some of the main industry organizations working on profile definitions and certifications for IoT constrained nodes and networks. Therearevarious documents and promotions from these organizations in the IoT space, so it is worth being familiar with themand their goals.

- **Internet Protocol for Smart Objects (IPSO) Alliance**

  Established in 2008, the Internet Protocol for Smart Objects (IPSO) Alliance has had its objective evolve over years. The alliance initially focused on promoting IP as the premier solution for smart objects communications. Today, it is more focused on how to use IP,with the IPSO Alliance organizing interoperability tests between alliance members to validate that IP for smart objects can work together and properly implement industry standards. The IPSO Alliance does not define technologies, as that is the role of the IETF and other standard organizations, but it documents the use of IP-based technologies for various IoT use cases and participates in educating the industry. As the IPSO Alliance declares in its value and mission statement, it wants to ensure that "engineers and product builders will have access to the necessary tools for 'how to build the IoTRIGHT.'"

- **Wi-SUN Alliance**

  The Wi-SUN Alliance is an example of efforts from the industry to define a communication profile that applies to specific physical and data link layer protocols. Currently, Wi-SUN's main focus is on the IEEE 802.15.4g protocol and its support for multiservice and secure IPv6 communications with applications running over the UDP transport layer. The utilities industry is the main area of focus for the Wi-SUN Alliance. The Wi-SUN field area network (FAN) profile enables smart utility networks to provide resilient, secure, and cost-effective connectivity with extremely good coverage in a range of topographic environments, from dense urban neighborhoods to ruralareas.

- **Thread**

  A group of companies involved with smart object solutions for consumers created the Thread Group. This group has defined an IPv6-based wireless profile that provides the best way to connect more than 250 devices into a low-power, wireless mesh network. The wireless technology used by Thread is IEEE 802.15.4, which is different from Wi-SUN's IEEE 802.15.4g

- **IPv6 Ready Logo**

Initially, the IPv6 Forum ensured the promotion of IPv6 around the world. Once IPv6 implementations became widely available, the need for interoperability and certification led to the creation of the IPv6 Ready Logo program. The IPv6 Ready Logo program has established conformance and interoperability testing programs with the intent of increasing user confidence when implementing IPv6. The IPv6 Core and specific IPv6 components, such as DHCP, IPsec, and customer edge router certifications, are in place. These certifications have industry-wide recognition, and many products are already certified. An IPv6 certification effort specific to IoT is currently under definition for the program.

3.5 Application Protocols for IoT

Application protocols that are sufficient for generic nodes and traditional networks often are not well suited for constrained nodes and networks. So here the focus is on how higher-layer IoT protocols are transported with following sections:

i. The Transport Layer: IP-based networks use either TCP or UDP. However, the constrained nature of IoT networks requires a closer look at the use of these traditional transport mechanisms.

ii. IoT Application Transport Methods: This section explores the various types of IoT application data and the ways this data can be carried across a network.

As in traditional networks, TCP or UDP are utilized in most cases when transporting IoT application data. With the lower-layer IoT protocols, there are typically multiple options and solutions presented for transporting IoT application data. This is because IoT is still developing and maturing and has to account for the transport of not only new application protocols and technologies but legacy ones as well.

3.6 The Transport Layer

This section reviews theselection of a protocol for the transport layer as supported by the TCP/IP architecture in the context of IoT networks. With the TCP/IP protocol, two main protocols are specified for the transport layer:

i. Transmission Control Protocol (TCP): This connection-oriented protocol requires a session to get established between the source and destination before exchanging data. It can be viewed equivalent to a traditional telephone conversation, in which two phones must be connected and the communication link established before the parties can talk.

i. User Datagram Protocol (UDP): With this connectionless protocol, data can be quickly sent between source and destination—but with no guarantee of delivery. This is analogous to the traditional mail delivery system, in which a letter is mailed to a destination. Confirmation of thereception of this letter does not happen until another letter is sent in response.

With the predominance of human interactions over the Internet, TCP is the main protocol used at the transport layer. This is largely due to its inherent characteristics, such as its ability to transport large volumes of data into smaller sets of packets. In addition, it ensures reassembly in a correct sequence, flow control and

window adjustment, and retransmission of lost packets. These benefits occur with the cost of overhead per packet and per session, potentially impacting overall packet per second performances and latency.

In contrast, UDP is most often used in the context of network services, such as Domain Name System (DNS), Network Time Protocol (NTP), Simple Network Management Protocol (SNMP), and Dynamic Host Control Protocol (DHCP), or for real-time data traffic, including voice and video over IP. In these cases, performance and latency are more important than packet retransmissions because re-sending a lost voice or video packet does not add value. When the reception of packets mustbe guaranteed error free, the application layer protocol takes care of that function.

When considering the choice of a transport layer by a given IoT application layer protocol, it is recommended to evaluate the impact of this choice on both the lower and upper layers of the stack. For example, most of the industrial application layer protocols, are implemented over TCP, while their specifications may offer support for both transport models. The reason for this is that often these industrial application layer protocols are older and were deployed when data link layers were often unreliable and called for error protection.

While the use of TCP may not strain generic compute platforms and high-data-rate networks, it can be challenging and is often overkill on constrained IoT devices and networks. This is particularly true when an IoT device needs to send only a few bytes of data per transaction. When using TCP, each packet needs to add a minimum of 20 bytes of TCP overhead, while UDP adds only 8 bytes. TCP also requires the establishment and potential maintenance of an open logical channel.

This may explain why a new IoT application protocol, such as Constrained Application Protocol (CoAP), almost always uses UDPand why implementations of industrial application layer protocols maycall for the optimization and adoption of the UDP transport layer if run over LLNs. For example, the Device Language Message Specification/Companion Specification for Energy Metering (DLMS/COSEM) application layer protocol, a popular protocol for reading smart meters in the utilities space, is the standard in Europe. Adjustments or optimizations to this protocol should be made depending on the IoT transport protocols that are present in the lower layers.

When transferring large amounts of DLMS/COSEM data, cellular links are preferred to optimize each open association. Smaller amounts of data can be handled efficiently over LLNs. Because packet loss ratios are generally higher on LLNs than on cellular networks, keeping the data transmission amounts small over LLNs limits the retransmission of large numbers of bytes.Multicast requirements are also impacted by the protocol selected for the transport layer. With multicast, a single message can be sent to multiple IoT devices. This is useful in the IoT context for upgrading the firmware of many IoT devices at once. Also, keep in mind that multicast utilizes UDP exclusively.

3.7 IoT Application Transport Methods

The following categories of IoT application protocols and their transport methods are explored in the following sections:

- **Application layer protocol not present:** In this case, the data payload is directly transported on top of the lower layers. No application layer protocol is used.
- **Supervisory control and data acquisition (SCADA):** SCADA is one of the most common industrial protocols in the world, but it was developed long before the days of IP, and it has been adapted for IP networks.
- **Generic web-based protocols:** Generic protocols, such as Ethernet, Wi-Fi, and 4G/LTE, are found on many consumer- and enterprise-class IoT devices that communicate over non- constrained networks.
- **IoT application layer protocols:** IoT application layer protocols are devised to run on constrained nodes with a small compute footprint and are well adapted to the network bandwidth constraints on cellular or satellite links or constrained 6LoWPAN networks. Message Queuing Telemetry Transport (MQTT) and Constrained Application Protocol (CoAP), are two examples of IoT application layer protocols.

3.7.1 Application Layer Protocol Not Present

IETF RFC 7228 devices defined as class 0 send or receive only a few bytes of data. For myriad reasons, such as processing capability, power constraints, and cost, these devices do not implement a fully structured network protocol stack, such as IP, TCP, or UDP, or even an application layer protocol. Class 0 devices are usually simple smart objects that are severely constrained. Implementing a robust protocol stack is usually not useful and sometimes not even possible with the limited available resources.

For example, consider low-cost temperature and relative humidity(RH) sensors sending data over an LPWA LoRaWAN infrastructure. Temperature is represented as 2 bytes and RH as another 2 bytes of data. Therefore, this small data payload is directly transported on top of the LoRaWAN MAC layer, without the use of TCP/IP. Example 3-1 shows the raw data for temperature and relative humidity and how it can be decoded by the application.

Example 3-1 *Decoding Temperature and Relative Humidity Sensor Data*

Temperature data payload over the network: Tx = 0x090c Temperature conversion required by the application

T = Tx/32 - 50

T = 0x090c/32 - 50

T = 2316/32 - 50 = **22.4°**

RH data payload over the network: RHx = 0x062e RH conversion required by the application:

100RH = RHx/16-24

100RH = 0x062e/16-24 = 74.9

RH **= 74.9%**

While many constrained devices, such as sensors and actuators, have adopted deployments that have no application layer, this transportation method has not been standardized. This lack of standardization makes it difficult for generic implementations of this transport method to be successful from an interoperability perspective.

Imagine expanding Example 3-1 to different kinds of temperature sensors from different manufacturers. These sensors will report temperature data in varying formats. A temperature value will always be present in the data transmitted by each sensor, but decoding this data will be vendor specific. If same scenario is scaled across hundreds or thousands of sensors, the problem of allowing various applications to receive and interpret temperature values delivered in different formats becomes increasingly complex. The solution to this problem is to use an IoT data broker, as detailed in Figure 3.11. An IoT data broker is a piece of middleware that standardizes sensor output into a common format that can then be retrieved by authorized applications.

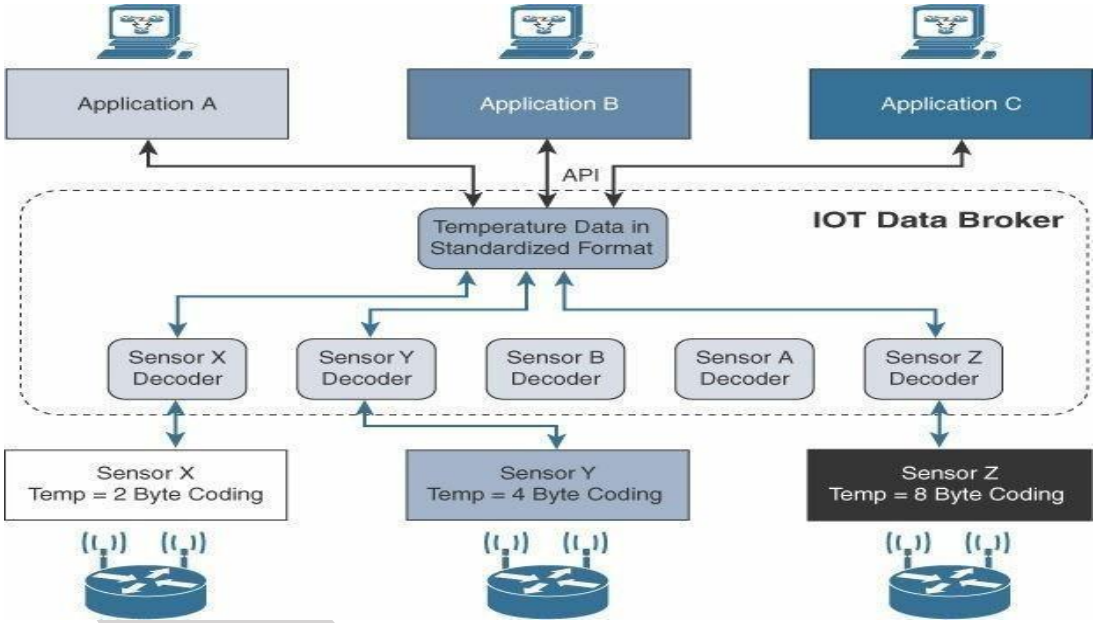


Figure 3.11 *IoT Data Broker*

In Figure 3.11, Sensors X, Y, and Z are all temperature sensors, but their output is encoded differently. The IoT data broker understands the different formats in which the temperature is encoded and is therefore able to decode this data into a common, standardized format. Applications A, B, and C in Figure 3.11 can access this temperature data without having to deal with decoding multiple temperature dataformats.

IoT data brokers are also utilized from a commercial perspective to distribute and sell IoT data to third parties. Companies can provide access to their data broker from another company's application for a fee. This makes an IoT data broker a possible revenue stream, depending on the value of the data it contains.

3.7.2 SCADA

In the world of networking technologies and protocols, IoT is relatively new. Combined with the fact that IP is the de facto standard for computer networking in general, older protocols that connected sensors and

actuators have evolved and adapted themselves to utilize IP.

A prime example of this evolution is **supervisory control and data acquisition (SCADA)**. Designed decades ago, SCADA is an automation control system that was initially implemented without IPover serial links, before being adapted to Ethernet and IPv4.

A Little Background on SCADA

For many years, vertical industries have developed communication protocols that fit their specific requirements. Many of them were defined and implemented when the most common networking technologies were serial link-based, such as RS-232 and RS-485. This led to SCADA networking protocols, which were well structured compared to the protocols described in the previous section, running directly over serial physical and data link layers.

At a high level, SCADA systems collect sensor data and telemetry from remote devices, while also providing the ability to control them. Used in today's networks, SCADA systems allow global, real- time, data-driven decisions to be made about how to improve business processes.

SCADA networks can be found across various industries, but SCADA is found mainly concentrated in theutilities and manufacturing/industrial verticals. Within these specific industries, SCADA commonlyuses certain protocols for communications between devices and applications. For example, Modbus and its variants are industrial protocols used to monitor and program remote devices via a master/slave relationship. Modbus is also found in building management, transportation, and energy applications. The DNP3 and International Electrotechnical Commission (IEC) 60870-5-101 protocols are found mainly in the utilities industry, along with DLMS/COSEM and ANSI C12 for advanced meter reading (AMR).

3.7.2.1 Adapting SCADA for IP

In the 1990s, the rapid adoption of Ethernet networks in the industrial world drove the evolution of SCADA application layer protocols. For example, the IEC adopted the Open SystemInterconnection (OSI) layer model to define its protocol framework. Other protocol user groups also slightly modified their protocols to run over an IP infrastructure. Benefits of this move to Ethernet and IP include the ability to leverage existing equipment and standards while integrating seamlessly the SCADA subnetworks to the corporateWAN infrastructures.

To further facilitate the support of legacy industrial protocols over IP networks, protocol specifications were updated and published, documenting the use of IP for each protocol. This included assigning TCP/UDP port numbers to the protocols, such as the following:

- DNP3 (adopted by IEEE 1815-2012) specifies the use of TCP or UDP on port 20000 for transporting DNP3 messages over IP.
- The Modbus messaging service utilizes TCP port 502.
- IEC 60870-5-104 is theevolution of IEC 60870-5-101 serial for running over Ethernet and IPv4 using port 2404.
- DLMS User Association specified a communication profile based on TCP/IP in the DLMS/COSEM Green Book (Edition 5 or higher), or in the IEC 62056-53 and IEC 62056-47 standards, allowing data exchange via IP and port 4059.

These legacy serial protocols have adapted and evolved to utilize IP and TCP/UDP as both networking

and transport mechanisms. This has allowed utilities and other companies to continue leveraging their investment in equipment and infrastructure, supporting these legacy protocols with modern IP networks. Let's dig deeper into how these legacy serial protocols have evolved to use IP by looking specifically at DNP3 as a representative use case.Like many of the other SCADA protocols, DNP3 is based on a master/slave relationship. The term *master* in this case refers to what is typically a powerful computer located in the control center of a utility, and a *slave* is a remote device with computing resources found in a location such as a substation. DNP3 refers to slaves specifically as *outstations*.

Outstations monitor and collect data from devices that indicate their state, such as whether a circuit breaker is on or off, and take measurements, including voltage, current, temperature, and so on. This data is then transmitted to the master when it is requested, or events and alarms can be sent in an asynchronous manner. The master also issues control commands, such as to start a motor or reset a circuit breaker, and logs the incoming data.

The IEEE 1815-2012 specification describes how the DNP3 protocol implementation must be adaptedto run either over TCP (recommended) or UDP. This specification defines connection management between the DNP3 protocol and the IP layers, as shown in Figure 3.12. Connection management links the DNP3 layers with the IP layers in addition to the configuration parameters and methods necessary for implementing the network connection. The IP layers appear transparent to the DNP3 layers as each piece of the protocol stack in one station logically communicates with therespective part in theother. This meansthat the DNP3 endpoints or devices are not aware of the underlying IP transport that is occurring.

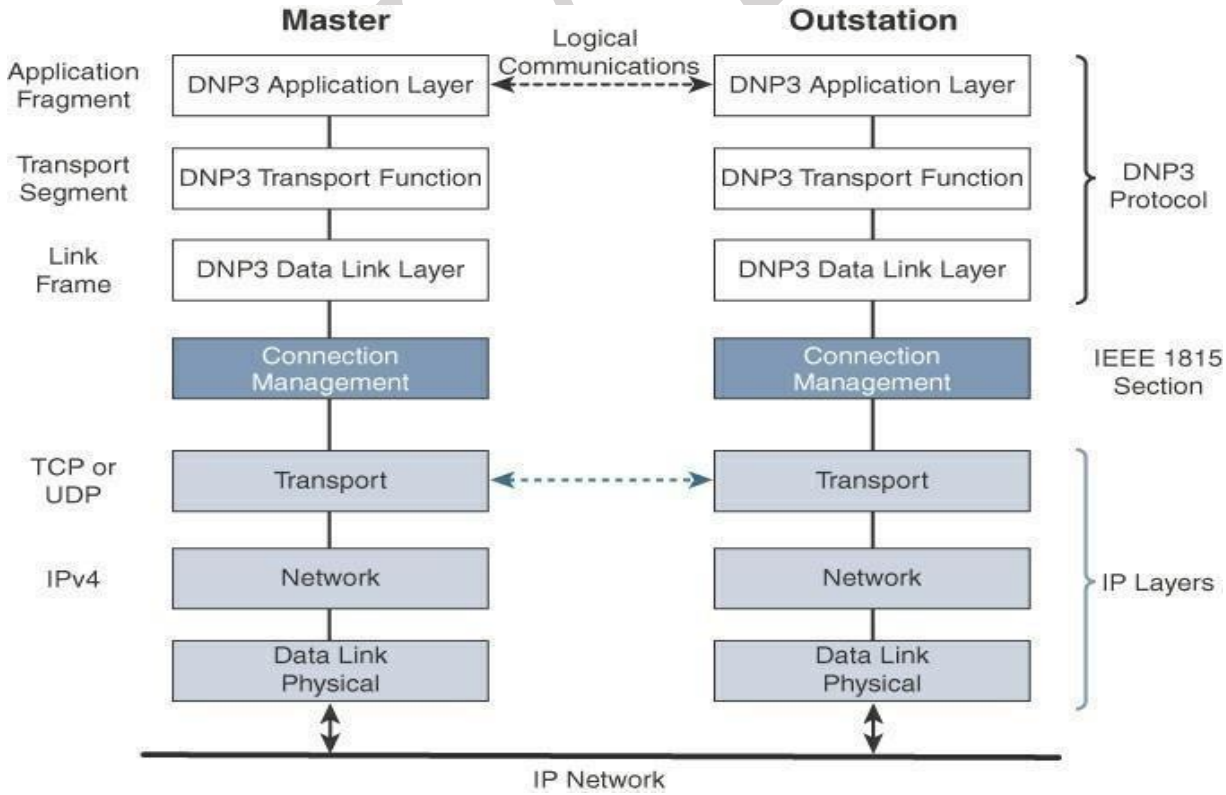


Figure 3.12 *Protocol Stack for Transporting Serial DNP3 SCADA over IP*

In Figure 3.12, the master side initiates connections by performing a TCP active open. The outstation listens for a connection request by performing a TCP passive open. *Dual endpoint* is defined as a process that can both listen for connection requests and perform an active open on the channel if required.

Master stations may parse multiple DNP3 data link layer frames from a single UDP datagram, while DNP3 data link layer frames cannot span multiple UDPdatagrams. Single or multiple connections to the master may get established while a TCP keepalive timer monitors the status of the connection. Keepalive messages are implemented as DNP3 data link layer status requests. If a response is not received to a keepalive message, the connection is deemed broken, and the appropriate action is taken.

3.7.2.2 Tunneling Legacy SCADA over IP Networks

Deployments of legacy industrial protocols, such as DNP3 and other SCADA protocols, in modern IP networks call for flexibility when integrating several generations of devices or operations that are tied to various releases and versions of application servers. Native support for IP can vary and mayrequire different solutions. Ideally, end-to-end native IP support is preferred, using a solution like IEEE 1815-2012 in the case of DNP3. Otherwise, transport of the original serial protocol over IP can be achieved either by tunneling using raw sockets over TCP or UDP or by installing an intermediate device that performs protocol translation between the serial protocol version and its IP implementation.

A raw socket connection simply denotes that the serial data is being packaged directly into a TCP or UDP transport. A socket in this instance is a standard application programming interface (API) composed of an IP address and a TCP or UDPport that is used to access network devices over an IP network. More modern industrial application servers may support this capability, while older versions typically require another device or piece of software to handle the transition from pure serial data to serial over IP using a raw socket. Figure 3.13 details raw socket scenarios for a legacy SCADA server trying to communicatewith remote serial devices.

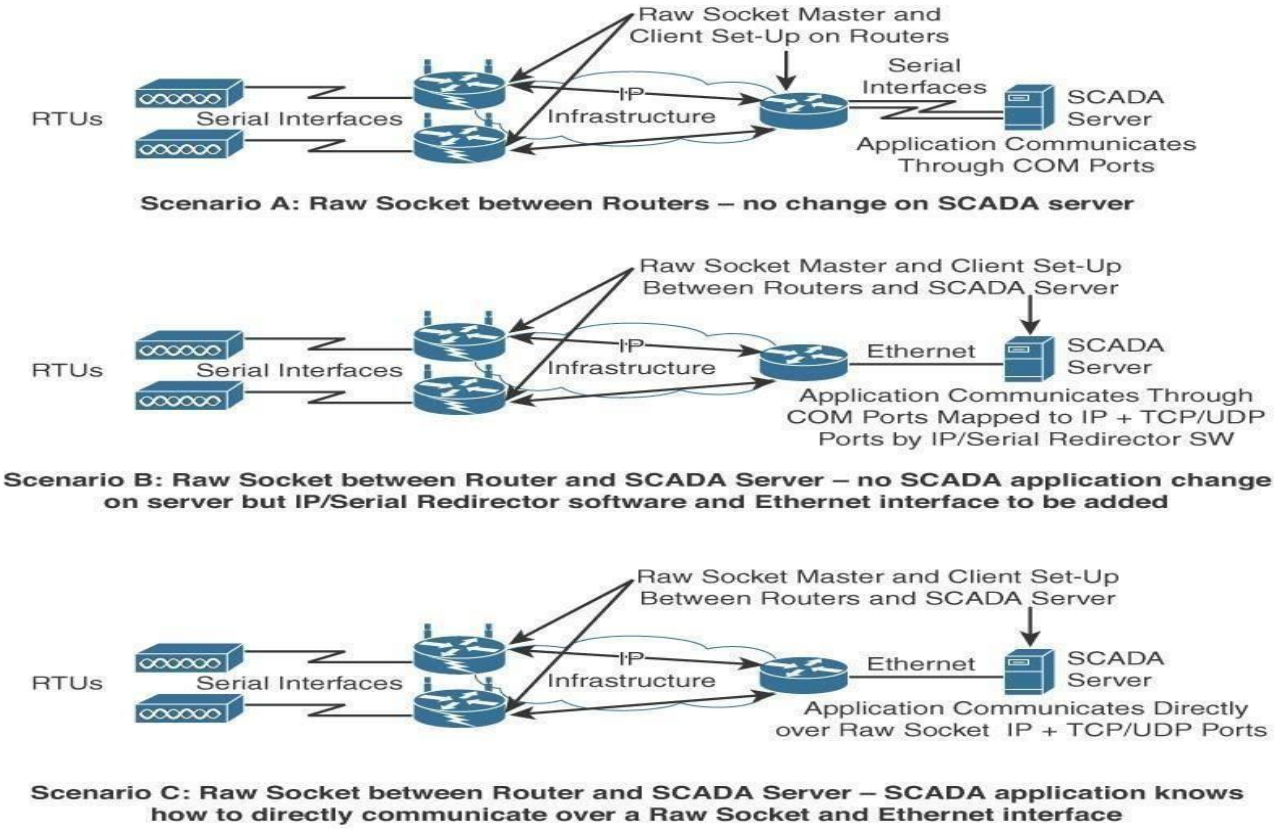


Figure 3.13 *Raw Socket TCP or UDP Scenarios for Legacy Industrial Serial Protocols*

In all the scenarios in Figure 3.13, notice that routers connect via serial interfaces to the remote terminal units (RTUs), which are often associated with SCADA networks. An RTU is a multipurpose device used to monitor and control various systems, applications, and devices managing automation. From the master/slave perspective, the RTUs are the slaves. Opposite the RTUsin each Figure 3.13 scenario is a SCADA server, or master, that varies its connection type. In reality, other legacy industrial application servers could be shown here as well.

Scenario A in Figure 3.13, both the SCADA server and the RTUs have a direct serial connection to their respective routers. The routers terminate the serial connections at both ends of the link and use raw socket encapsulation to transport the serial payload over the IP network.

Scenario B has a small change on the SCADA server side. A piece of software is installed on the SCADA server that maps the serial COM ports to IP ports. This software is commonly referred to as an IP/serial redirector. The IP/serial redirector in essence terminates the serial connection of the SCADA server and converts it to a TCP/IP port using a raw socket connection.

Scenario C in Figure 3.13, the SCADA server supports native raw socket capability. Unlike in Scenarios A and B, where a router or IP/serial redirector software has to map the SCADA server's serial ports to IP ports, in Scenario C the SCADA server has full IP support for raw socket connections.

3.7.2.3 SCADA Protocol Translation

As mentioned earlier, an alternative to a raw socket connection for transporting legacy serial data across an IP network is protocol translation. With protocol translation, the legacy serial protocol is translated to a corresponding IP version. For example, Figure 3.14 shows two serially connected DNP3 RTUs and two master applications supporting DNP3 over IP that control and pull data from the RTUs. The IoT gateway in this figure performs a protocol translation function that enables communication between the RTUs and servers, despite the fact that a serial connection is present on one side and an IP connection is used on the other.

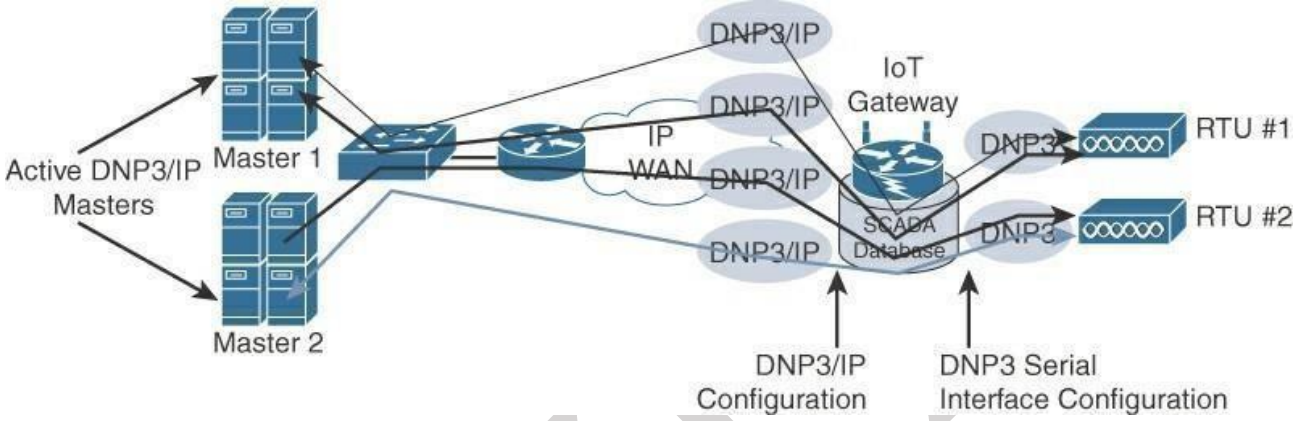


Figure 3.14 *DNP3 Protocol Translation*

By running protocol translation, the IoT gateway connected to the RTUs in Figure 3.14 is implementing a computing function close to the edge of the network. Adding computing functions close to the edge helps scale distributed intelligence in IoT networks. This can be accomplished by offering computing resources on IoT gateways or routers, as shown in this protocol translation example. Alternatively, this can also be performed directly on a node connecting multiple sensors. In either case, this is referred to as fog computing.

3.7.2.4 SCADA Transport over LLNs with MAP-T

Due to the constrained nature of LLNs, the implementation of industrial protocols should at a minimum be done over UDP. This in turn requires that both the application servers and devices support and implement UDP. While the long-term evolution of SCADA and other legacy industrial protocols is to natively support IPv6, it must be highlighted that most, if not all, of the industrial devices supporting IP today support IPv4 only. When deployed over LLN subnetworks that are IPv6 only, a transition mechanism, such as MAP-T (Mapping of Address and Port using Translation, RFC 7599), needs to be implemented. This allows the deployment to take advantage of native IPv6 transport transparently to the application and devices.

Figure 3.15 depicts a scenario in which a legacy endpoint is connected across an LLN running 6LoWPAN to

an IP-capable SCADA server. The legacy endpoint could be running various industrial and SCADA protocols, including DNP3/IP, Modbus/TCP, or IEC 60870-5-104. In this scenario, the legacy devices and the SCADA server support only IPv4 (typical in the industry today). However, IPv6 (with 6LoWPAN and RPL) is being used for connectivity to the endpoint. 6LoWPANis a standardized protocol designed for constrained networks, but it only supports IPv6. In this situation, the end devices, the endpoints, and the SCADA server support only IPv4, but the network in the middle supports onlyIPv6.

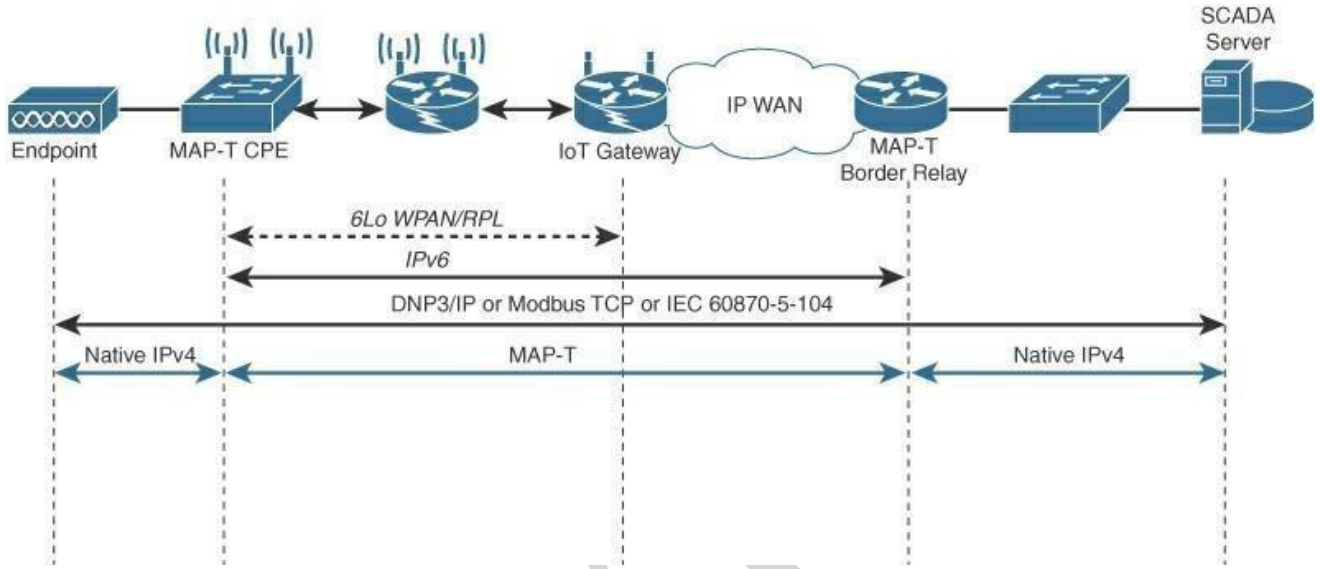


Figure 3.15 *DNP3 Protocol over 6LoWPAN Networks with MAP-T*

The solution to this problem is to use the protocol known as MAP-T, MAP-T makes the appropriate mappings between IPv4 and the IPv6 protocols. This allows legacy IPv4 traffic to be forwarded across IPv6 networks. In other words, older devices and protocols can continue running IPv4 even though the network is requiring IPv6.

In Figure 3.15 the IPv4 endpoint on the left side is connected to a Customer Premise Equipment (CPE) device. The MAP-T CPE device has an IPv6 connection to the RPL mesh. On the right side, a SCADA server with native IPv4 support connects to a MAP-T border gateway. The MAP-T CPE device and MAP-T border gateway are thus responsible for the MAP-T conversion from IPv4 to IPv6.

Legacy implementations of SCADA and other industrial protocols are still widely deployed across many industries. While legacy SCADA has evolved from older serial connections to support IP, still it can be expected to see mixed deployments for many years. To address this challenge, OT networks require mechanisms such as raw sockets and protocol translation to transport legacy versions over modern IP networks. Even when the legacy devices have IPv4 capability, the constrained portions ofthe network often require IPv6, not IPv4. In these cases, a MAP-T solution can be put in place to enable IPv4 data to be carried across an IPv6 network.

3.7.3 Generic Web-Based Protocols

Over the years, web-based protocols have become common in consumer and enterprise applications and services. Therefore, it makes sense to try to leverage these protocols when developing IoT applications, services, and devices in order to ease the integration of data and devices from prototyping to production.The level of familiarity with generic web-based protocols is high. Therefore, programmers with basic web programming skills can work on IoT applications, and this may lead to innovative ways to deliver and handle real-time IoT data. For example, an IoT device generating an event can have the result of launching a video capture, while at the same time a notification is sent to a collaboration tool, such as a Cisco Spark room. This notification allows technicians and engineers to immediately start working on this alert. In addition to a generally high level of familiarity with web-based protocols, scaling methods for web environments are also well understood—and this is crucial when developing consumer applications for potentially large numbers of IoT devices.

Once again, the definition of constrained nodes and networks must be analyzed to select the most appropriate protocol. On non- constrained networks, such as Ethernet, Wi-Fi, or 3G/4G cellular, where bandwidth is not perceived as a potential issue, data payloads based on a verbose data model representation, including XML or JavaScript Object Notation (JSON), can be transported over HTTP/HTTPS or WebSocket. This allows implementers to develop their IoT applications in contexts similar to web applications.

The HTTP/HTTPS client/server model serves as the foundation for the World Wide Web. Recent evolutions of embedded web server software with advanced features are now implemented with very little memory (in the range of tens of kilobytes in some cases). This enables the use of embedded web services software on some constrained devices.

When considering web services implementation on an IoT device, the choice between supporting the client or server side of the connection must be carefully weighed. IoT devices that only push data to an application (for example, an Ethernet- or Wi-Fi-based weather station reporting data to a weather map application or a Wi-Fi–enabled body weight scale that sends data to a health application) may need to implement web services on the client side. The HTTP client side only initiates connections and does not accept incoming ones.

On the other hand, some IoT devices, such as a video surveillance camera, may have web services implemented on the server side. However, because these devices often have limited resources, the number of incoming connections must be kept low. In addition, advanced development in data modeling should be considered as a way to shift the workload from devices to clients, including web browsers on PCs, mobile phones, tablets, and cloud applications.

Interactions between real-time communication tools powering collaborative applications, such as voice and video, instant messaging, chat rooms, and IoT devices, are also emerging. This is driving the need for simpler communication systems between people and IoT devices. One protocol that addresses this need is Extensible Messaging and Presence Protocol (XMPP).

3.7.4 IoT Application Layer Protocols

Whenconsidering constrained networks and/or a large-scale deployment of constrained nodes, verbose web-based and data model protocols, as discussed in the previous section, maybe too heavy for IoT applications. To address this problem, the IoT industry is working on new lightweight protocols that are better suited to large numbers of constrained nodes and networks. Two of the most popular protocols are CoAP and MQTT. Figure 3.16 highlights their position in a common IoT protocol stack.

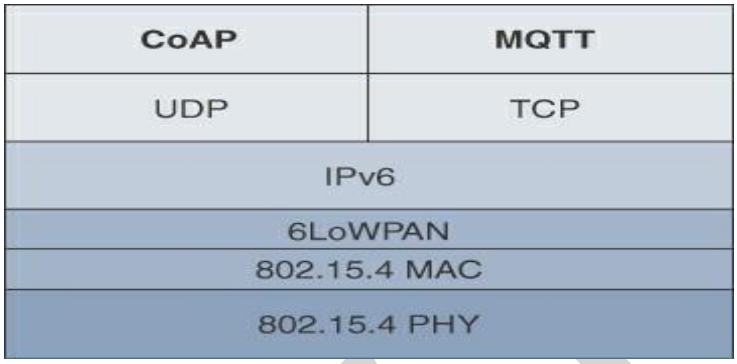


Figure 3.16 *Example of a High-Level IoT Protocol Stack for CoAP and MQTT*

In Figure 3.16, CoAP and MQTT are naturally at the top of this sample IoT stack, based on an IEEE 802.15.4 mesh network. While there are a few exceptions, like CoAP deployed over UDP and MQTT running over TCP. The following sections take a deeper look at CoAP and MQTT.

3.7.4.1 CoAP

Constrained Application Protocol (CoAP) resulted from the IETF Constrained RESTful Environments (CoRE) working group's efforts to develop a generic framework for resource-oriented applications targeting constrained nodes and networks.

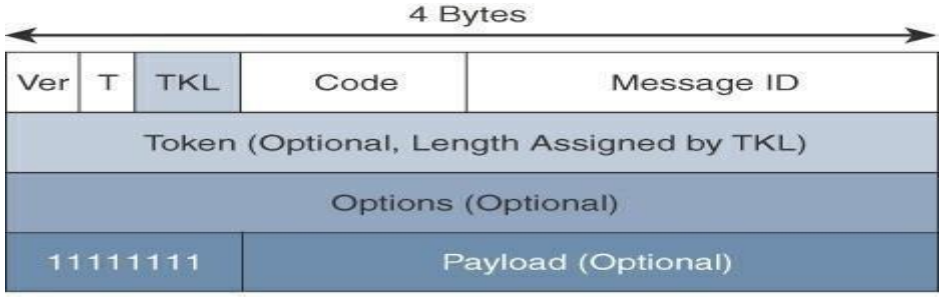
The CoAP framework defines simple and flexible ways to manipulate sensors and actuators for data or device management. The IETF CoRE working group has published multiple standards-track specifications for CoAP, including the following:

- **RFC 6690:** Constrained RESTful Environments (CoRE) Link Format
- **RFC 7252:** The Constrained Application Protocol (CoAP)
- **RFC 7641:** Observing Resources in the Constrained Application Protocol (CoAP)
- **RFC 7959:** Block-Wise Transfers in the Constrained Application Protocol (CoAP)
- **RFC 8075:** Guidelines for Mapping Implementations: HTTP to the Constrained Application Protocol (CoAP)

The CoAP messaging model is primarily designed to facilitate the exchange of messages over UDP between endpoints, including the secure transport protocol Datagram Transport Layer Security (DTLS). The IETF CoRE working group is studying alternate transport mechanisms, including TCP, secure TLS, and WebSocket. CoAP over Short Message Service (SMS) as defined in Open Mobile Alliance for Lightweight Machine-to-

Machine (LWM2M) for IoT device management is also being considered.

RFC 7252 provides more details on securing CoAP with DTLS. It specifies how a CoAP endpoint is provisioned with keys and a filtering list. Four security modes are defined: NoSec, PreSharedKey, RawPublicKey, and Certificate. The NoSec and RawPublicKey implementations are mandatory. From a formatting perspective, a CoAP message is composed of a short fixed-length Header field (4 bytes), a variable-



length but mandatory Token field (0–8 bytes), Options fields if necessary, and the Payload field. Figure 3.17 details the CoAP message format, which delivers low overhead while decreasing parsing complexity.

Figure 3.17 *CoAP Message Format*

From Figure 3.17, the CoAP message format is relatively simple and flexible. It allows CoAP to deliver low overhead, which is critical for constrained networks, while also being easy to parse and process for constrained devices. Table 6-1 provides an overview of the various fields of a CoAP message.

Table 6-1 *CoAP Message Fields*

CoAP Message Field	Description
Ver (Version)	Identifies the CoAP version.
T (Type)	Defines one of the following four message types: Confirmable (CON), Non-confirmable (NON), Acknowledgement (ACK), or Reset (RST). CON and ACK are highlighted in more detail in Figure 6-9.
TKL (Token Length)	Specifies the size (0–8 Bytes) of the Token field.
Code	Indicates the request method for a request message and a response code for a response message. For example, in Figure 6-9, GET is the request method, and 2.05 is the response code. For a complete list of values for this field, refer to RFC 7252.
Message ID	Detects message duplication and used to match ACK and RST message types to Con and NON message types.
Token	With a length specified by TKL, correlates requests and responses.
Options	Specifies option number, length, and option value. Capabilities provided by the Options field include specifying the target resource of a request and proxy functions.
Payload	Carries the CoAP application data. This field is optional, but when it is present, a single byte of all 1s (0xFF) precedes the payload. The purpose of this byte is to delineate the end of the Options field and the beginning of Payload.

CoAP can run over IPv4 or IPv6. However, it is recommended that the message fit within a single IP packet and UDP payload to avoid fragmentation. For IPv6, with the default MTU size being 1280 bytes and allowing for no fragmentation across nodes, themaximum CoAP message size could be up to 1152 bytes, including 1024 bytes for the payload. In the case of IPv4, as IP fragmentation may exist across the network, implementations should limit themselves to more conservative values and set the IPv4 Don't Fragment (DF)bit.

While most sensor and actuator traffic utilizes small-packet payloads, some use cases, such as firmware upgrades, require the capability to send larger payloads. CoAP doesn't rely on IP fragmentation but defines (in RFC 7959) a pair of Block options for transferring multiple blocks of information from a resource representation in multiple request/response pairs.

As illustrated in Figure 3.18, CoAP communications across an IoT infrastructure can take various paths. Connections can be between devices located on thesame or different constrained networks or between devices and generic Internet or cloud servers, all operating over IP. Proxy mechanisms are also defined, and RFC 7252 details a basic HTTP mapping for CoAP. As both HTTP and CoAP are IP-based protocols, the proxy function can be located practically anywhere in the network, not necessarily at the border between constrained and non-constrained networks.

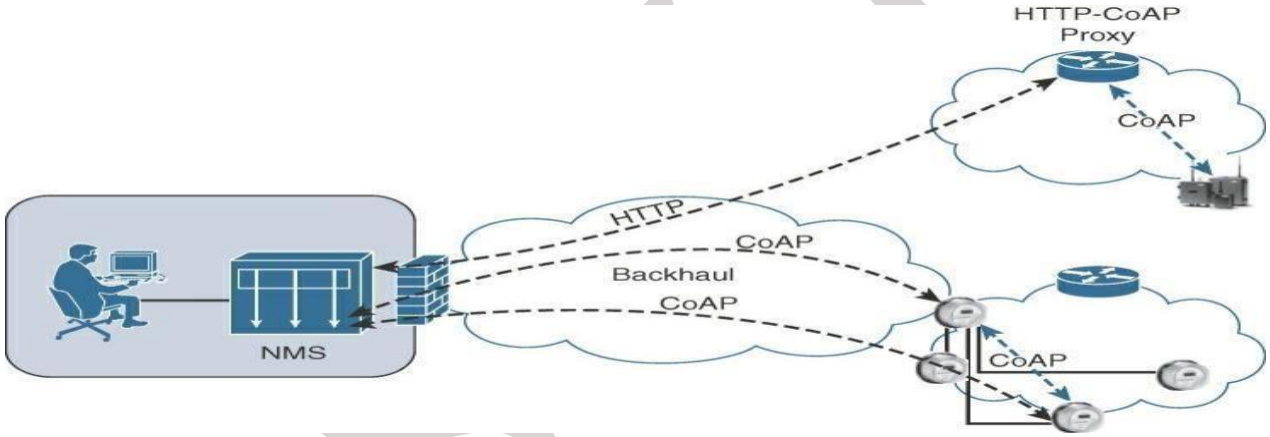


Figure 3.18 *CoAP Communications in IoT Infrastructures*

Just like HTTP, CoAP is based on the REST architecture, but with a "thing" acting as both the client and the server. Through the exchange of asynchronous messages, a client requests an action via a method code on a server resource. A uniform resource identifier (URI) localized on theserver identifies this resource. The server responds with a response code that mayinclude a resource representation. The CoAP request/response semantics include the methods GET, POST, PUT, and DELETE.

CoAP defines four types of messages: confirmable, non-confirmable, acknowledgement, and reset. Method codes and response codes included in some of these messages makethem carry requests or responses. CoAP code, method and response codes, option numbers, and content format have been assigned by IANA as Constrained RESTful Environments (CoRE) parameters.

While running over UDP, CoAP offers a reliable transmission of messages when a CoAP header is marked as "confirmable." In addition, CoAP supports basic congestion control with a default time-out, simple stop and wait retransmission with exponential back-off mechanism, and detection of duplicate messages through a

message ID. If a request or response is tagged as confirmable, the recipient must explicitly either acknowledge or reject the message, using the same message ID, as shown in Figure 3.19. If a recipient can't process a non-confirmable message, a reset message is sent.

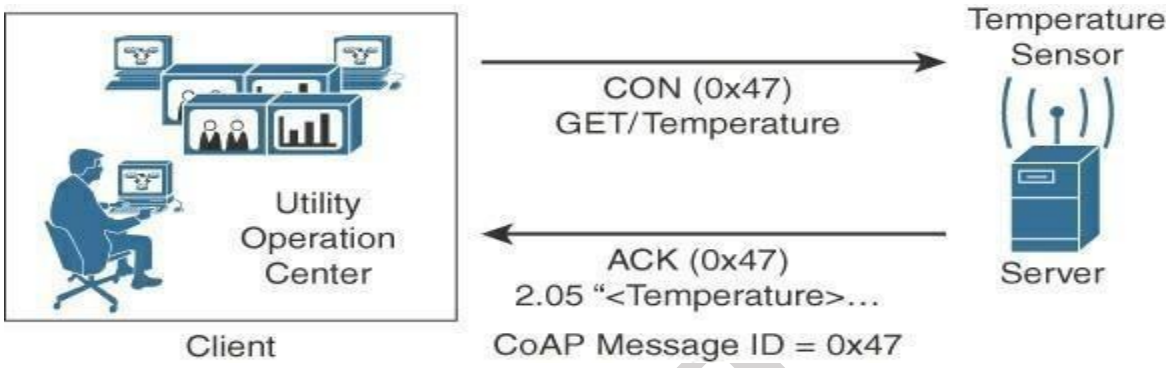


Figure 3.19 *CoAP Reliable Transmission Example*

Figure 3.19 shows a utility operations center on the left, acting as the CoAP client, with the CoAP server being a temperature sensor on the right of the figure. The communication between the client and server uses a CoAP message ID of 0x47. The CoAP Message ID ensures reliability and is used to detect duplicate messages. The client in Figure 3.19 sends a GET message to get the temperature from the sensor. Notice that the 0x47 message ID is present for this GET message and that the message is also marked with CON. A CON, or confirmable, marking in a CoAP message means the message will be retransmitted until the recipient sends an acknowledgement (or ACK) with the same message ID.

In Figure 3.19, the temperature sensor does reply with an ACK message referencing the correct message ID of 0x47. In addition, this ACK message piggybacks a successful response to the GET request itself. This is indicated by the 2.05 response code followed by the requested data. CoAP supports data requests sent to a group of devices by leveraging the use of IP Multicast. Implementing IP Multicast with CoAP requires the use of all-CoAP-node multicast addresses. For IPv4 this address is 224.0.1.187, and for IPv6 it is FF0X::FD. These multicast addresses are joined by CoAP nodes offering services to other endpoints while listening on the default CoAP port, 5683. Therefore, endpoints can find available CoAP services through multicast service discovery. A typical use case for multicasting is deploying a firmware upgrade for a group of IoT devices, such as smart meters.

With often no affordable manual configuration on the IoT endpoints, a CoAP server offering services and resources needs to be discovered by the CoAP clients. Services from a CoAP server can either be discovered by learning a URI in a namespace or through the "All CoAP nodes" multicast address. When utilizing the URI scheme for discovering services, the default port 5683 is used for non-secured CoAP, or **coap**, while port 5684 is utilized for DTLS-secured CoAP, or **coaps**. The CoAP server must be in listening state on these ports, unless a different port number is associated with the URI in a namespace. Much as with accessing web server resources, CoAP specifications provide a description of the relationships between

resources in RFC 6690, "Constrained RESTful Environments (CoRE) Link Format."

To improve the response time and reduce bandwidth consumption, CoAP supports caching capabilities based on the response code. To use a cache entry, a CoAP endpoint must validate the presented request and stored response matches, including all options (unless marked as NoCacheKey). This confirms that the stored response is fresh or valid. A wide range of CoAP implementations are available. Some are published with open source licenses, and others are part of vendor solutions

3.7.4.2 Message Queuing Telemetry Transport (MQTT)

At the end of the 1990s, engineers from IBM and Arcom (acquired in 2006 byEurotech) were looking for a reliable, lightweight, and cost-effective protocol to monitor and control a large number of sensors and their data from a central server location, as typically used by the oil and gas industries.
Their research resulted in the development and implementation of the Message Queuing Telemetry Transport (MQTT) protocol that is now standardized by the Organization for the Advancement of Structured Information Standards (OASIS).

Considering the harsh environments in the oil and gas industries, an extremely simple protocol with only a few options was designed, with considerations for constrained nodes, unreliable WAN backhaul communications, and bandwidth constraints with variable latencies. These were some of the rationales for the selection of a client/server and publish/subscribe framework based on the TCP/IP architecture, as shown in Figure 6-10.

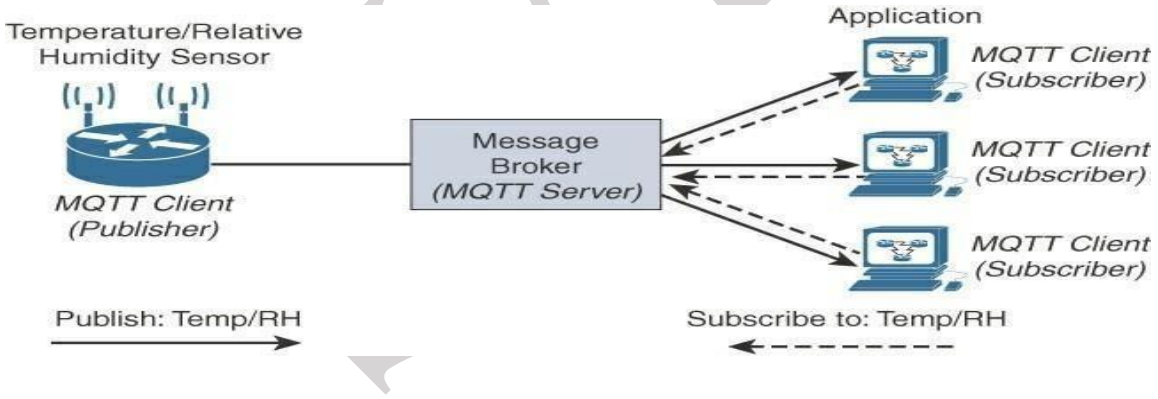


Figure 3.20 *MQTT Publish/Subscribe Framework*

An MQTT client can act as a publisher to send data (or resource information) to an MQTT server acting as an MQTT message broker. In the example illustrated in Figure 3.20, the MQTT client on the left side is a temperature (Temp) and relative humidity (RH) sensor that publishes its Temp/RH data. The MQTT server (or message broker) accepts the network connection along with application messages, such as Temp/RH data, from the publishers. It also handles the subscription and unsubscription process and pushes the application data to MQTT clients acting as subscribers.

The application on the right side of Figure 3.20 is an MQTT client that is a subscriber to the Temp/RH data being generated by the publisher or sensor on the left. This model, where subscribers express a desire to

receive information from publishers, is well known. A great example is the collaboration and social networking application Twitter.

With MQTT, clients can subscribe to all data (using a wildcard character) or specific data from the information tree of a publisher. In addition, the presence of a message broker in MQTT decouples the data transmission between clients acting as publishers and subscribers. In fact, publishers and subscribers do not even know (or need to know) about each other. A benefit of having this decoupling is that the MQTT message broker ensures that information can be buffered and cached in case of network failures. This also means that publishers and subscribers do not have to be online at the sametime.

MQTT control packets run over a TCP transport using port 1883. TCP ensures an ordered, lossless stream of bytes between the MQTT client and the MQTT server. Optionally, MQTT can be secured using TLS on port 8883, and WebSocket (defined in RFC 6455) can also be used.

MQTT is a lightweight protocol because each control packet consists of a 2-byte fixed header with optional variable header fields and optional payload. Control packet can contain a payload up to 256 MB.

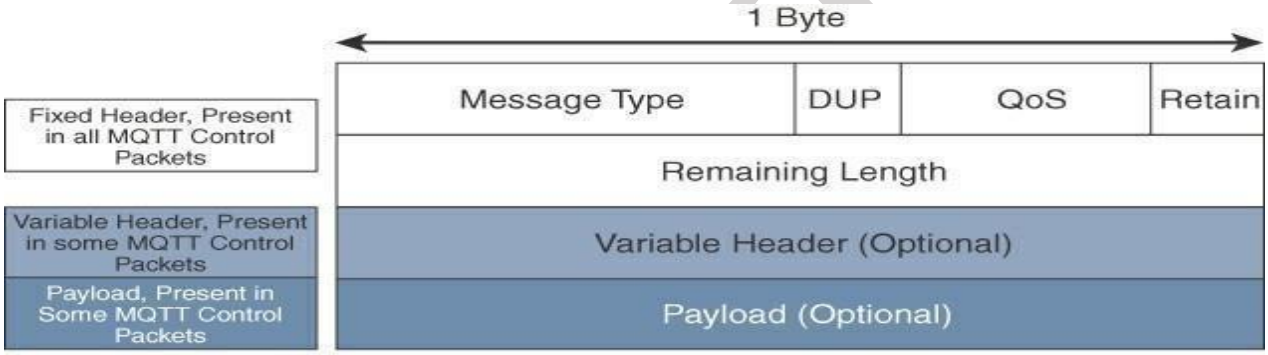


Figure 3.21 *MQTT Message Format*

Compared to the CoAP message format in Figure 3.17, MQTT contains a smaller header of 2 bytes compared to 4 bytes for CoAP. The first MQTT field in the header is Message Type, which identifies the kind of MQTT packet within a message. Fourteen different types of control packets are specified in MQTT version 3.1.1. Each of them has a unique value that is coded into the Message Type field. Note that values 0 and 15 are reserved. MQTT message types are summarized in Table3.2.

The next field in the MQTT header is DUP (Duplication Flag). This flag, when set, allows the client to notate that the packet has been sent previously, but an acknowledgement was not received. The QoS header field allows for the selection of three different QoS levels. The next field is the Retain flag. Only found in a PUBLISH message, the Retain flag notifies the server to hold onto the message data. This allows new subscribers to instantly receive the last known value without having to wait for the next update from the publisher. The last mandatory field in the MQTT message header is Remaining Length. This field specifies the number of bytes in the MQTT packet following thisfield.

Table 6-2 *MQTT Message Types*

Message Type	Value	Flow	Description
CONNECT	1	Client to server	Request to connect
CONNACK	2	Server to client	Connect acknowledgement
PUBLISH	3	Client to server Server to client	Publish message
PUBACK	4	Client to server Server to client	Publish acknowledgement
PUBREC	5	Client to server Server to client	Publish received
PUBREL	6	Client to server Server to client	Publish release
PUBCOMP	7	Client to server Server to client	Publish complete
SUBSCRIBE	8	Client to server	Subscribe request
SUBACK	9	Server to client	Subscribe acknowledgement
UNSUBSCRIBE	10	Client to server	Unsubscribe request
UNSUBACK	11	Server to client	Unsubscribe acknowledgement
PINGREQ	12	Client to server	Ping request
PINGRESP	13	Server to client	Ping response
DISCONNECT	14	Client to server	Client disconnecting

The MQTT protocol offers three levels of quality of service (QoS). QoS for MQTT is implemented when exchanging application messages with publishers or subscribers, and it is different fromthe IP QoS that most people are familiar with. The delivery protocol is symmetric. This means the client and server can each take the role of either sender or receiver. The delivery protocol is concerned solely with the delivery of an application message from a single sender to a single receiver. These are the three levels of MQTTQoS:

- **QoS 0:** This is a best-effort and unacknowledged data service referred to as “at most once” delivery. The publisher sends its message one time to a server, which transmits it once to the subscribers. No response is sent by the receiver, and no retry is performed by the sender. The message arrives at the receiver either once or not at all.
- **QoS 1:** This QoS level ensures that the message delivery between the publisher and server and then between the server and subscribers occurs at least once. In PUBLISH and PUBACK packets, a packet identifier is included in thevariable header. If the message is not acknowledged by a PUBACKpacket, it is sent again. This level guarantees “at least once” delivery.
- **QoS 2:** This is the highest QoS level, used when neither loss nor duplication of messages is acceptable. There is an increased overhead associated with this QoS level because each packet contains an optional variable header with a packet identifier. Confirming the receipt of a PUBLISH message requires a two-step acknowledgement process. The first step is done through the PUBLISH/PUBREC packet pair, and the second is achieved with the PUBREL/PUBCOMP packet pair. This level provides a “guaranteed service” known as “exactly once” delivery, with no consideration for the number of retries as long as the message is delivered once.

As mentioned earlier, the QoS process is symmetric in regard to the roles of sender and receiver, but two separate transactions exist. One transaction occurs between the publishing client and the MQTT server, and the other transaction happens between the MQTT server and the subscribing client. Figure 3.22 provides an overview of the MQTT QoS flows for the three different levels.

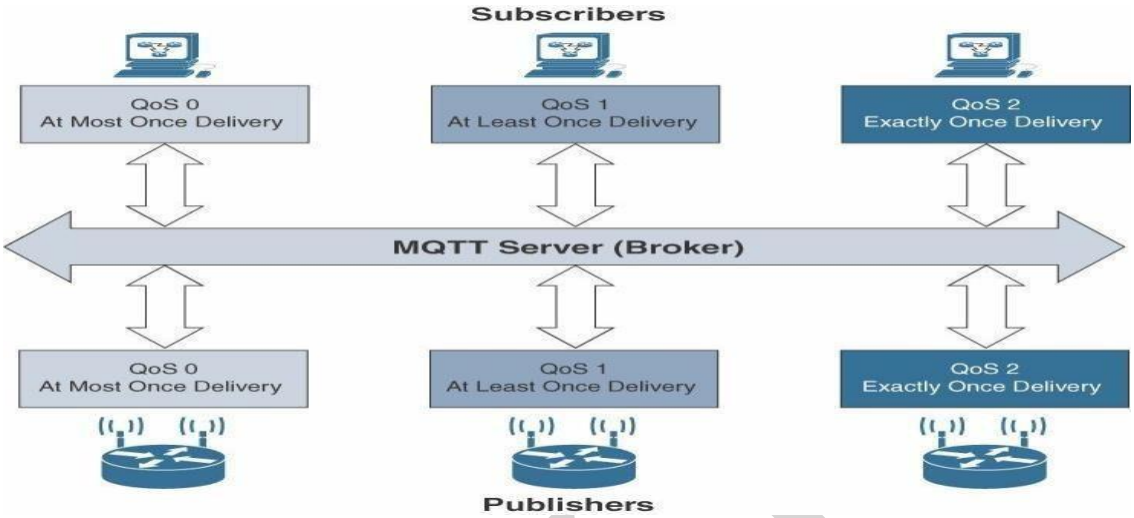


Figure 3.22 *MQTT QoS Flows*

As with CoAP, a wide range of MQTT implementations are now available. They are either published as open source licenses or integrated into vendors' solutions, such as Facebook Messenger.

Both CoAP and MQTT have been discussed in detail, there arises questions like "Which protocol is better for a given use case?" and "Which one should I used in my IoT network?" Unfortunately, the answer is not always clear, and both MQTT and CoAP have their place. Table 3-3 provides an overview of the differences between MQTT and CoAP, along with their strengths and weaknesses from an IoT perspective.

Table 3-3 *Comparison Between CoAP and MQTT*

Factor	CoAP	MQTT
Main transport protocol	UDP	TCP
Typical messaging	Request/response	Publish/subscribe
Effectiveness in LLNs	Excellent	Low/fair (Implementations pairing UDP with MQTT are better for LLNs.)
Security	DTLS	SSL/TLS
Communication model	One-to-one	many-to-many
Strengths	Lightweight and fast, with low overhead, and suitable for constrained networks; uses a RESTful model that is easy to code to; easy to parse and process for constrained devices; support for multicasting; asynchronous and synchronous messages	TCP and multiple QoS options provide robust communications; simple management and scalability using a broker architecture
Weaknesses	Not as reliable as TCP-based MQTT, so the application must ensure reliability.	Higher overhead for constrained devices and networks; TCP connections can drain low-power devices; no multicasting support

MODULE -4
Data and Analytics for IoT

4.1 An Introduction to Data Analytics for IoT

In the world of IoT, the creation of massive amounts of data from sensors is common and one of the biggest challenges—not only from a transport perspective but also from a data management standpoint. A great example of the deluge of data that can be generated by IoT is found in the commercial aviation industry and the sensors that are deployed throughout an aircraft.

Modern jet engines are fitted with thousands of sensors that generate a whopping 10GB of data per second. For example, modern jet engines, similar to the one shown in Figure 7-1, maybe equipped with around 5000 sensors. Therefore, a twin engine commercial aircraft with these engines operating on average 8 hours a day will generate over 500 TB of data daily, and this is just the data from the engines! Aircraft today have thousands of other sensors connected to the airframe and other systems. In fact, a single wing of a modern jumbo jet is equipped with 10,000 sensors.

Figure 7-1 *Commercial Jet Engine*

The potential for a petabyte (PB) of data per day per commercial airplane is not farfetched—and this is just for *one* airplane. Across the world, there are approximately 100,000 commercial flights per day. The amount of IoT data coming just from the commercial airline business is overwhelming.

4.1.1 Structured Versus Unstructured Data

Structured data and unstructured data are important classifications as they typically require different toolsets from a data analytics perspective. Figure 7-2 provides a high-level comparison of structured data and unstructured data.

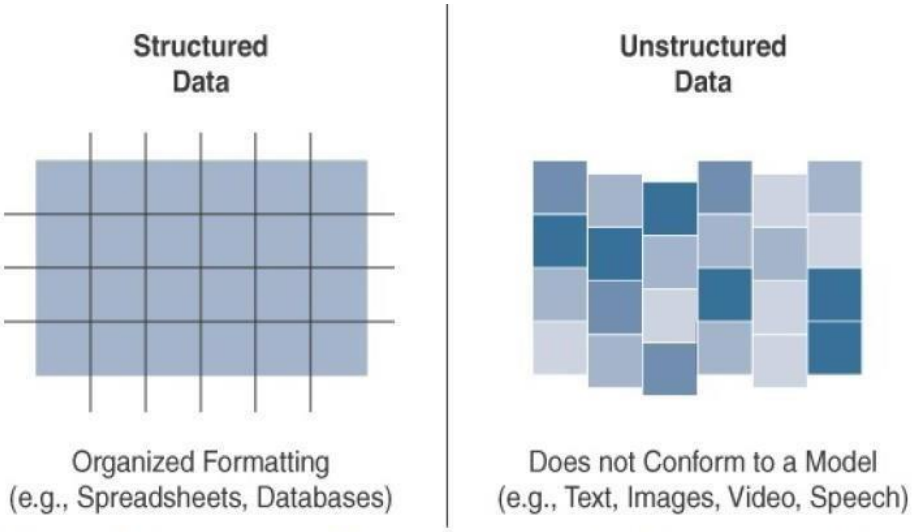


Figure 7-2 *Comparison Between Structured and Unstructured Data*

Structured data means that the data follows a model or schema that defines how the data is represented or organized, meaning it fits well with a traditional relational database management system (RDBMS). In many cases you will find structured data in a simple tabular form—for example, a spreadsheet where data occupies a specific cell and can be explicitly defined and referenced.

Structured data can be found in most computing systems and includes everything from banking transaction and invoices to computer log files and router configurations. IoT sensor data often uses structured values, such as temperature, pressure, humidity, and so on, which are all sent in a known format. Structured data is easily formatted, stored, queried, and processed; for these reasons, it has been the core type of data used for making business decisions.

Unstructured data lacks a logical schema for understanding and decoding the data through traditional programming means. Examples of this data type include text, speech, images, and video. As a general rule, any data that does not fit neatly into a predefined data model is classified as unstructured data.

According to some estimates, around 80% of a business's data is unstructured. Because of this fact, data analytics methods that can be applied to unstructured data, such as cognitive computing and machine learning, are deservedly garnering a lot of attention. With machine learning applications, such as natural language processing (NLP), you can decode speech. With image/facial recognition applications, you can extract critical information from still images and

video.Data in Motion Versus Data at Rest

As in most networks, data in IoT networks is either in transit ("data in motion") or being held or stored ("data at rest"). Examples of data in motion include traditional client/server exchanges, such as web browsing and file transfers, and email. Data saved to a hard drive, storage array, or USB drive is data at rest.

From an IoT perspective, the data from smart objects is considered data in motion as it passes through the network en route to its final destination. This is often processed at the edge, using fog computing. When data is processed at the edge, it may be filtered and deleted or forwarded on for further processing and possible storage at a fog node or in the data center. Data does not come to rest at the edge.

Data at rest in IoT networks can be typically found in IoT brokers or in some sort of storage array at the data center. Myriad tools, especially tools for structured data in relational databases, are available from a data analytics perspective. The best known of these tools is Hadoop. Hadoop not only helps with data processing but also data storage.

4.1.2 IoT Data Analytics Overview

The true importance of IoT data from smart objects is realized only when the analysis of the data leads to actionable business intelligence and insights. Data analysis is typically broken down by the types of results that are produced. As shown in Figure 7-3, there are four types of data analysis results:

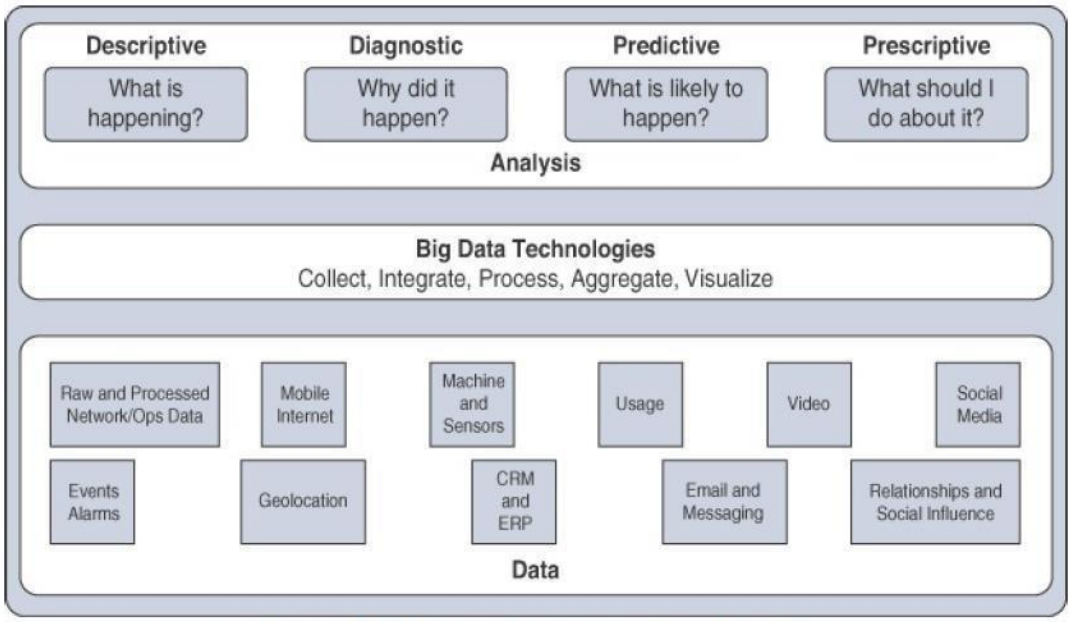


Figure 7-3 *Types of Data Analysis Results*

4.1.2.1 **Descriptive:** Descriptive data analysis tells you what is happening, either now or in the past. For example, a thermometer in a truck engine reports temperature values every second. From a descriptive analysis perspective, you can pull this data at any moment to gain insight into the current operating condition of the truck engine. If the temperature value is too high, then there may be a cooling problem or the engine may be experiencing too much load.

4.1.2.2 **Diagnostic:** When you are interested in the "why," diagnostic data analysis can provide the answer. Continuing with the example of the temperature sensor in the truck engine, you might wonder why the truck engine failed. Diagnostic analysis might show that the temperature of the engine was too high, and the engine overheated. Applying diagnostic analysis across the data generated by a wide range of smart objects can provide a clear picture of why a problem or an event occurred.

4.1.2.3 **Predictive:** Predictive analysis aims to foretell problems or issues before they occur. For example, with historical values of temperatures for the truck engine, predictive analysis could provide an estimate on the remaining life of certain components in the engine. These components could then be proactively replaced before failure occurs. Or perhaps if temperature values of the truck engine start to rise slowly over time, this could indicate the need for an oil change or some other sort of engine cooling maintenance.

4.1.2.4 **Prescriptive:** Prescriptive analysis goes a step beyond predictive and

recommends solutions for upcoming problems. A prescriptive analysis of the temperature data from a truck engine might calculate various alternatives to cost-effectively maintain our truck. These calculations could range from the cost necessary for more frequent oil changes and cooling maintenance to installing new cooling equipment on the engine or upgrading to a lease on a model with a more powerful engine. Prescriptive analysis looks at a variety of factors and makes the appropriate recommendation.

Both predictive and prescriptive analyses are more resource intensive and increase complexity, but the value they provide is much greater than the value from descriptive and diagnostic analysis. Figure 7-4 illustrates the four data analysis types and how they rank as complexity and value increase. You can see that descriptive analysis is the least complex and at the same time offers the least value. On the other end, prescriptive analysis provides the most value but is the most complex to implement. Most data analysis in the IoT space relies on descriptive and diagnostic analysis, but a shift toward predictive and prescriptive analysis is understandably occurring for most businesses and organizations.

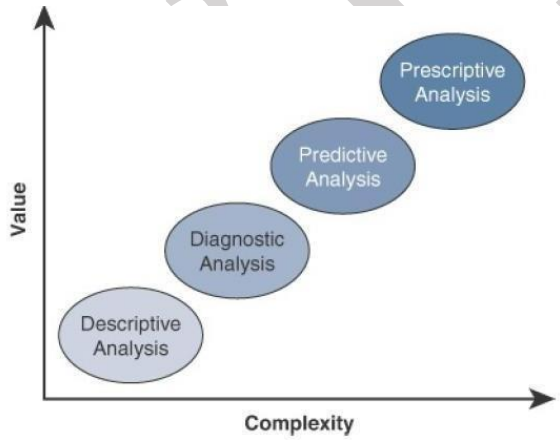


Figure 7-4 *Application of Value and Complexity Factors to the Types of Data Analysis*

As IoT has grown and evolved, it has become clear that traditional data analytics solutions were not always adequate. For example, traditional data analytics typically employs a standard RDBMS and corresponding tools, but the world of IoT is much more demanding. While relational databases are still used for certain data types and applications, they often struggle with the nature of IoT data. IoT data places two specific challenges on a relational database:

4.1.2.5 **Scaling problems:** Due to the large number of smart objects in most IoT networks that continually send data, relational databases can grow incredibly large very quickly. This can result in performance issues that can be costly to

resolve, often requiring more hardware and architecture changes.

4.1.2.6 **Volatility of data:** With relational databases, it is critical that the schema be designed correctly from the beginning. Changing it later can slow or stop the database from operating. Due to the lack of flexibility, revisions to the schema must be kept at a minimum. IoT data, however, is volatile in the sense that the data model is likely to change and evolve over time. A dynamic schema is often required so that data model changes can be made daily or even hourly.

To deal with challenges like scaling and data volatility, a different type of database, known as NoSQL, is being used. Structured Query Language (SQL) is the computer language used to communicate with an RDBMS. As the name implies, a NoSQL database is a database that does not use SQL. It is not set up in the traditional tabular form of a relational database. NoSQL databases do not enforce a strict schema, and they support a complex, evolving data model. These databases are also inherently much more scalable.

4.2 Machine Learning

ML is indeed central to IoT. Data collected by smart objects needs to be analyzed, and intelligent actions need to be taken based on these analyses. Performing this kind of operation manually is almost impossible (or very, very slow and inefficient).

Machines are needed to process information fast and react instantly when thresholds are met. For example, every time a new advance is made in the field of self-driving vehicles, abnormal pattern recognition in a crowd, or any other automated intelligent and machine-assisted decision system, ML is named as the tool that made the advance possible. But ML is not new. It was invented in the middle of the twentieth century and actually fell out of fashion in the 1980s

4.2.1 Machine Learning Overview

ML is concerned with any process where the computer needs to receive a set of data that is processed to help perform a task with more efficiency. ML is a vast field but can be simply divided in two main categories: supervised and unsupervised learning.

Supervised Learning

In supervised learning, the machine is trained with input for which there is a known correct answer. For example, suppose that you are training a system to recognize when there is a human in a mine tunnel. A sensor equipped with a basic camera can capture shapes and return them to a computing system that is responsible for determining whether the shape is a human or something else (such as a vehicle, a pile of ore, a rock, a piece of wood, and so on.). With supervised learning techniques,

hundreds or thousands of images are fed into the machine, and each image is labeled (human or nonhuman in this case). This is called the *training set*. An algorithm is used to determine common parameters and common differences between the images. The comparison is usually done at the scale of the entire image, or pixel by pixel. Images are resized to have the same characteristics

(resolution, color depth, position of the central figure, and so on), and each point is analyzed. Human images have certain types of shapes and pixels in certain locations (which correspond to the position of the face, legs, mouth, and so on). Each new image is compared to the set of known "good images," and a deviation is calculated to determine how different the new image is from the average human image and, therefore, the probability that what is shown is a human figure. This process is called *classification*.

After training, the machine should be able to recognize human shapes. Before real field deployments, the machine is usually tested with unlabeled pictures— this is called the validation or the test set, depending on the ML system used—to verify that the recognition level is at acceptable thresholds. If the machine does not reach the level of success expected, more training is needed.

Unsupervised Learning

In some cases, supervised learning is not the best method for a machine to help with a human decision. Suppose that you are processing IoT data from a factory manufacturing small engines. You know that about 0.1% of the produced engines on average need adjustments to prevent later defects, and your task is to identify them before they get mounted into machines and shipped away from the factory. With hundreds of parts, it may be very difficult to detect the potential defects, and it is almost impossible to train a machine to recognize issues that may not be visible. However, you can test each engine and record multiple parameters, such as sound, pressure, temperature of key parts, and so on. Once data is recorded, you can graph these elements in relation to one another (for example, temperature as a function of pressure, sound versus rotating speed over time). You can then input this data into a computer and use mathematical functions to find groups. For example, you may decide to group the engines by the sound they make at a given temperature. A standard function to operate this grouping, *K-means clustering*, finds the mean values for a group of engines (for example, mean value for temperature, mean frequency for sound). Grouping the engines this way can quickly reveal several types of engines that all belong to the same category (for example, small engine of chainsaw type, medium engine of lawnmower type). All engines of the same type produce sounds and temperatures in the same range as the other members of the same group.

There will occasionally be an engine in the group that displays unusual characteristics (slightly out of expected temperature or sound range). This is the engine that you send formanual evaluation. The computing process associated with this determination is called *unsupervised learning*. This type of learning is unsupervised because there is not a "good" or "bad" answer known in advance. It is the variation from a group behavior that allows the computer to learn that something is different. The example of engines is, of course, very simple. In most cases, parameters are multidimensional. In other words, hundreds or thousands of parameters are computed, and

small cumulated deviations in multiple dimensions are used to identify the exception. Figure 7-5 shows an example of such grouping and deviation identification logic. Three parameters are graphed (components 1, 2, and 3), and four distinct groups (clusters) are found. You can see some points that are far from the respective groups. Individual devices that display such "out of cluster" characteristics should be examined more closely individually.

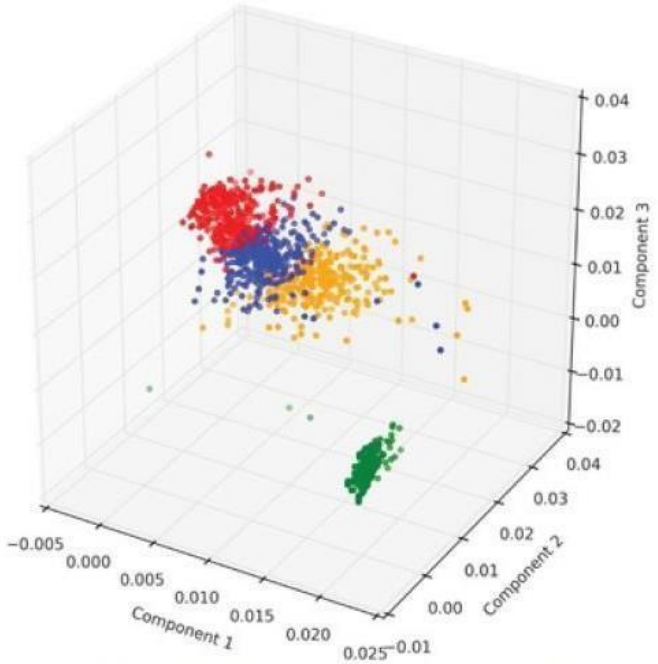


Figure 7-5 *Clustering and Deviation Detection Example*

Neural Networks

Distinguishing between a human and a car is easy. The computer can recognize that humans have distinct shapes (such as legs or arms) and that vehicles do not. Distinguishing a human from another mammal is much more difficult (although nonhuman mammals are not common occurrences in mines). The same goes for telling the difference between a pickup truck and a van. You can tell when you

see one, but training a machine to differentiate them requires more than basic shape recognition. This is where neural networks come into the picture. Neural networks are ML methods that mimic the way the human brain works. When you look at a human figure, multiple zones of your brain are activated to recognize colors, movements, facial expressions, and so on. Your brain combines these elements to conclude that the shape you are seeing is human. Neural networks mimic the same logic. The information goes through different algorithms (called *units*), each of which is in charge of processing an aspect of the information. The resulting value of one unit computation can be used directly or fed into another unit for further processing to occur. In this case, the neural

network is said to have several layers. For example, a neural network processing human image recognition may have two units in a first layer that determines whether the image has straight lines and sharp angles—because vehicles commonly have straight lines and sharp angles, and human figures do not. If the image passes the first layer successfully (because there are no or only a small percentage of sharp angles and straight lines), a second layer may look for different features (presence of face, arms, and so on), and then a third layer might compare the image to images of various animals and conclude that the shape is a human (or not). The great efficiency of neural networks is that each unit processes a simple test, and therefore computation is quite fast. This model is demonstrated in Figure 7-6.

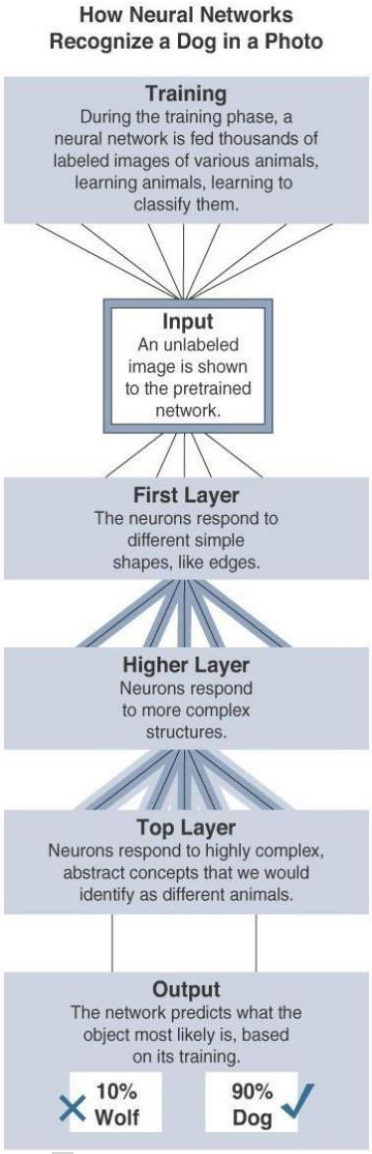


Figure 7-6 *Neural Network Example*

By contrast, old supervised ML techniques would compare the human figure to potentially hundreds of thousands of images during the training phase, pixel by pixel, making them difficult and expensive to implement (with a lot of training needed) and slow to operate. Neural networks have been the subject of much research work. Multiple research and optimization efforts have examined the number of units and layers, the type of data processed at each layer, and the type and combination of algorithms used to process the data to make processing more efficient for specific applications. Image processing can be optimized with certain types of algorithms that may not be optimal for crowd movement classification. Another algorithm may be found in this case that would revolutionize the way these movements are processed and analyzed. Possibilities are as numerous as the applications where they can be used.

4.2.2 Machine Learning and Getting Intelligence from Big Data

When the principles of machine learning are clear, the application to IoT becomes obvious. The difficulty resides in determining the right algorithm and the right learning model for each use case. Such an analysis goes beyond the scope of this chapter, but it can be useful to organize ML operations into two broad subgroups:

4.2.2.1 **Local learning:** In this group, data is collected and processed locally, either in the sensor itself (the edge node) or in the gateway (the fog node).

4.2.2.2 **Remote learning:** In this group, data is collected and sent to a central computing unit (typically the data center in a specific location or in the cloud), where it is processed.

Regardless of the location where (and, therefore, the scale at which) data is processed, common applications of ML for IoT revolve around four major domains:

4.2.2.3 **Monitoring:** Smart objects monitor the environment where they operate. Data is processed to better understand the conditions of operations. These conditions can refer to external factors, such as air temperature, humidity, or presence of carbon dioxide in a mine, or to operational internal factors, such as the pressure of a pump, the viscosity of oil flowing in a pipe, and so on. ML can be used with monitoring to detect early failure conditions (for example, K-means deviations showing out-of-range behavior) or to better evaluate the environment (such as shape recognition for a robot automatically sorting material or picking goods in a warehouse or a supply chain).

4.2.2.4 **Behavior control:** Monitoring commonly works in conjunction with behavior control. When a given set of parameters reach a target threshold — defined in advance (that is, supervised) or learned dynamically through deviation from mean values (that is, unsupervised)—monitoring functions generate an alarm. This alarm can be relayed to a human, but a more efficient and more advanced system would trigger a corrective action, such as increasing the flow of fresh air in the mine tunnel, turning the robot arm, or reducing the oil pressure in the pipe.

4.2.2.5 **Operations optimization:** Behavior control typically aims at taking corrective actions based on thresholds. However, analyzing data can also lead to changes that improve the overall process. For example, a water purification plant in a smart city can implement a system to monitor the efficiency of the purification process based on which chemical (from company A or company B) is used, at what temperature, and associated to what stirring mechanism (stirring speed and depth). Neural networks can combine multiples of such

units, in one or several layers, to estimate the best chemical and stirring mix for a target air temperature. This intelligence can help the plant reduce its consumption of chemicals while still operating at the same purification efficiency level. As a result of the learning, behavior control results in different machine actions. The objective is not merely to pilot the operations but to improve the efficiency and the result of these operations.

4.2.2.6 **Self-healing, self-optimizing:** A fast-developing aspect of deep learning is the closed loop. ML-based monitoring triggers changes in machine behavior (the change is monitored by humans), and operations optimizations. In turn, the ML engine can be programmed to dynamically monitor and combine new parameters (randomly or semi-randomly) and automatically deduce and implement new optimizations when the results demonstrate a possible gain. The system becomes self-learning and self-optimizing. It also detects new K-means deviations that result in predilection of new potential defects, allowing the system to self-heal. The healing is not literal, as external factors (typically human operators) have to intervene, but the diagnosis is automated. In many cases, the system can also automatically order a piece of equipment that is detected as being close to failure or automatically take corrective actions to avoid the failure (for example, slow down operations, modify a machine's movement to avoid fatigue on a weak link).

4.2.3 Predictive Analytics

Multiple smart objects measure the pull between carriages, the weight on each wheel, and multiple other parameters to offer a form of cruise control optimization for the driver. At the same time, cameras observe the state of the tracks ahead, audio sensors analyze the sound of each wheel on the tracks, and multiple engine parameters are measured and analyzed. All this data can be returned to a data processing center in the cloud that can re-create a virtual twin of each locomotive. Modeling the state of each locomotive and combining this knowledge with anticipated travel and with the states (and detected failures) of all other locomotives of the same type circulating on the tracks of the entire city, province, state, or country allows the analytics platform to make very accurate predictions on what issue is likely to affect each train and each locomotive. Such predictive analysis allows preemptive maintenance and increases the safety and efficiency of operations.

Big Data Analytics Tools and Technology

Generally, the industry looks to the "three *V*s" to categorize big data:

- **Velocity:** *Velocity* refers to how quickly data is being collected and analyzed. Hadoop Distributed File System is designed to ingest and process data very quickly. Smart objects

can generate machine and sensor data at a very fast rate and require database or file systems capable of equally fast ingest functions.

- **Variety:** *Variety* refers to different types of data. Often you see data categorized as structured, semi-structured, or unstructured. Different database technologies may only be capable of accepting one of these types. Hadoop is able to collect and store all threetypes. This can be beneficial when combining machine data from IoT devices that is very structured in nature with data from other sources, such as social media or multimedia that is unstructured.
- **Volume:** *Volume* refers to the scale of the data. Typically, this is measured from gigabytes on the very low end to petabytes or even exabytes of data on the other extreme. Generally, big data implementations scale beyond what is available on locally attached storage disks on a single node. It is common to see clusters of servers that consist of dozens, hundreds, or even thousands of nodes for some large deployments.

The characteristics of big data can be defined by the sources and types of data. First is machine data, which is generated by IoT devices and is typically unstructured data. Second istransactional data, which is from sources that produce data from transactions on these systems, and, have high volume and structured. Third is social data sources, which are typically high volume and structured. Fourth is enterprise data, which is data that is lower in volume and very structured. Hence big data consists of data from all these separate sources.

Massively Parallel Processing Databases

Massively parallel processing (MPP) databases were built on the concept of the relational data warehouses but are designed to be much faster, to be efficient, and to support reduced query times. To accomplish this, MPP databases take advantage of multiple nodes (computers) designed in a scale-out architecture such that both data and processing are distributed across multiple systems.

MPPs are sometimes referred to as *analytic databases* because they are designed to allow for fast query processing and often have built-in analytic functions. As the name implies, these database types process massive data sets in parallel across many processors and nodes. An MPP architecture (see Figure 7-7) typically contains a single master node that is responsible for the coordination of all the data storage and processing across the cluster. It operates in a "sharednothing" fashion, with each node containing local processing, memory, and storage and operating independently. Data storage is optimized across the nodes in a structured SQL-likeformat that allows data analysts to work with the data using common SQL tools and applications. The earlier example of a complex SQL query could be distributed and optimized, resulting in a significantly faster response. Because data stored on MPPs must still conform to this relational structure, it may not be the only database type used in an IoT implementation. The sources and types of data may vary, requiring a database that is more flexible than relational

databases allow.

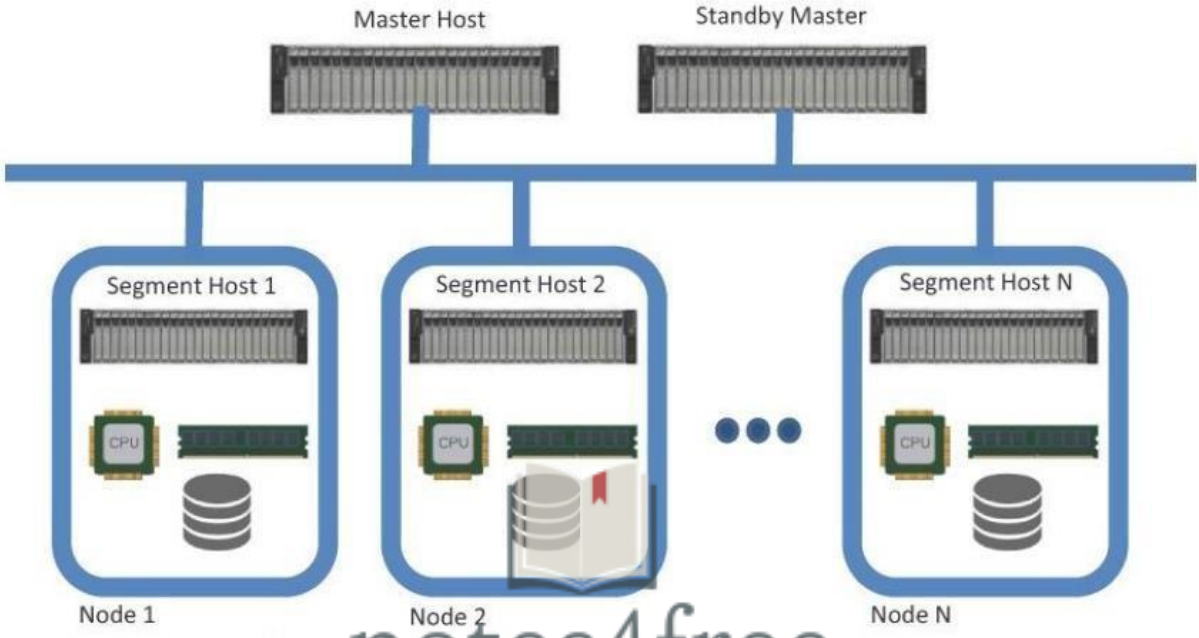


Figure 7-7 *MPP Shared-Nothing Architecture*

NoSQL Databases

NoSQL ("not only SQL") is a class of databases that support semi-structured and unstructured

data, in addition to the structured data handled by data warehouses and MPPs. NoSQL is not a specific database technology; rather, it is an umbrella term that encompasses several different types of databases, including the following:

- **Document stores:** This type of database stores semi-structured data, such as XML or JSON. Document stores generally have query engines and indexing features that allowfor many optimized queries.
- **Key-value stores:** This type of database stores associative arrays where a key is paired with an associated value. These databases are easy to build and easy to scale.
- **Wide-column stores:** This type of database stores similar to a key-value store, but the formatting of the values can vary from row to row, even in the same table.
- **Graph stores:** This type of database is organized based on the relationships between elements. Graph stores are commonly used for social media or natural language processing, where the connections between data are very relevant. NoSQL was developed to support the high-velocity, urgent data requirements of modern web applications that typically do not require much repeated use. The original intent was to quickly ingest rapidly changing server logs and clickstream data generated by web-scale applications that did not neatly fit

into the rows and columns required by relational databases. Similar to otherdata stores, like MPPs and Hadoop (discussed later), NoSQL is built to scale horizontally, allowing the database to span multiple hosts, and can even be distributed geographically.

Expanding NoSQL databases to other nodes is similar to expansion in other distributed data systems, where additional hosts are managed by a master node or process. This expansion can be automated by some NoSQL implementations or can be provisioned manually. This level of flexibility makes NoSQL a good candidate for holding machine and sensor data associated with smart objects.

Hadoop

Hadoop is the most recent entrant into the data management market, but it is arguably the most popular choice as a data repository and processing engine.

Hadoop was originally developed as a result of projects at Google and Yahoo!, and the original intent for Hadoop was to index millions of websites and quickly return search results for open source search engines. Initially, the project had two key elements:

- **Hadoop Distributed File System (HDFS):** A system for storing data across multiple nodes
- **MapReduce:** A distributed processing engine that splits a large task into smaller ones that can be run in parallel.

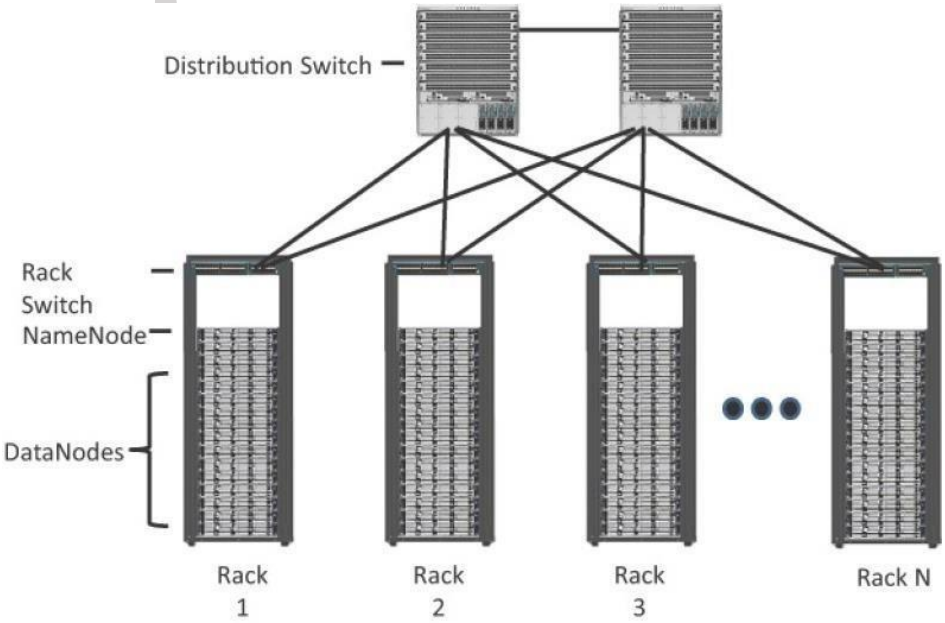


Figure 7-8 *Distributed Hadoop Cluster*

Much like the MPP and NoSQL systems discussed earlier, Hadoop relies on a scale-out architecture that leverages local processing, memory, and storage to distribute tasks and provide a scalable storage system for data. Both MapReduce and HDFS take advantage of this distributed architecture to store and process massive amounts of data and are thus able to leverage resources from all nodes in the cluster.

For HDFS, this capability is handled by specialized nodes in the cluster, including NameNodes and DataNodes (see Figure 7-8):

- **NameNodes:** These are a critical piece in data adds, moves, deletes, and reads on HDFS. They coordinate where the data is stored, and maintain a map of where each block of data is stored and where it is replicated. All interaction with HDFS is coordinated through the primary (active) NameNode, with a secondary (standby) NameNode notified of the changes in the event of a failure of the primary. The NameNode takes write requests from clients and distributes those files across the available nodes in configurable block sizes, usually 64 MB or 128 MB blocks. The NameNode is also responsible for instructing the DataNodes where replication should occur.
- **DataNodes**: These are the servers where the data is stored at the direction of the NameNode. It is common to have many DataNodes in a Hadoop cluster to store the data. Data blocks are distributed across several nodes and often are replicated three, four, or more times across nodes for redundancy. Once data is written to one of the DataNodes, the DataNode selects two (or more) additional nodes, based on replication policies, to ensure data redundancy across the cluster. Disk redundancy techniques such as Redundant Array of Independent Disks (RAID) are generally not used for HDFS because the NameNodesand DataNodes coordinate blocklevel redundancy with this replication technique. Figure 7-9 shows the relationship between NameNodes and DataNodes and how data blocks are distributed across the cluster.

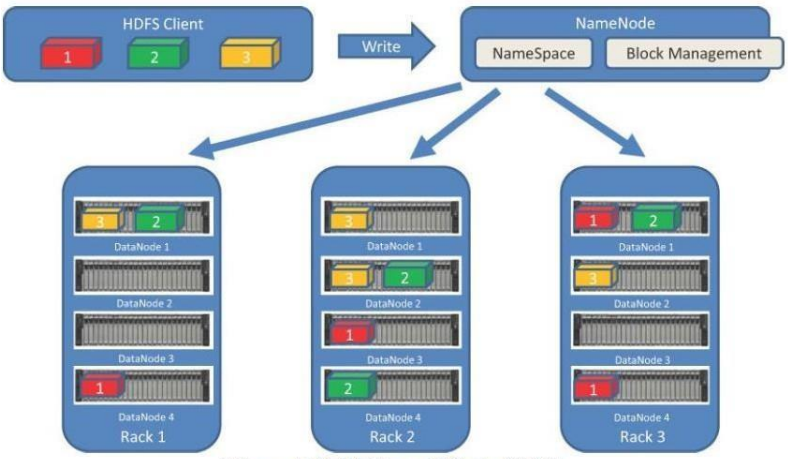


Figure 7-9 *Writing a File to HDFS*

MapReduce leverages a similar model to batch process the data stored on the cluster nodes. Batch processing is the process of running a scheduled or ad hoc query across historical data stored in the HDFS. A query is broken down into smaller tasks and distributed across all the nodes running MapReduce in a cluster. While this is useful for understanding patterns and trending in historical sensor or machine data, it has one significant drawback: time

YARN

Introduced with version 2.0 of Hadoop, YARN (Yet Another Resource Negotiator) was designed to enhance the functionality of MapReduce. With the initial release, MapReduce was responsible for batch data processing and job tracking and resource management across the cluster. YARN was developed to take over the resource negotiation and job/task tracking, allowing MapReduce to be responsible only for data processing.

With the development of a dedicated cluster resource scheduler, Hadoop was able to add additional data processing modules to its core feature set, including interactive SQL andreal-time processing, in addition to batch processing using MapReduce.

The Hadoop Ecosystem

As mentioned earlier, Hadoop plays an increasingly big role in the collection, storage, and processing of IoT data due to its highly scalable nature and its ability to work with large volumes of data.
Hadoop now comprises more than 100 software projects under the Hadoop umbrella, capable of nearly every element in the data lifecycle, from collection, to storage, to processing, to analysis and visualization. Each of these individual projects is a unique piece of the overall data management solution. The following sections describe several of these packages and discuss how they are used to collect or process data.

Apache Kafka

Part of processing real-time events, such as those commonly generated by smart objects, is having them ingested into a processing engine. The process of collecting data from a sensor or log file and preparing it to be processed and analyzed is typically handled by messaging systems. Messaging systems are designed to accept data, or messages, from where the data is generated and deliver the data to stream-processing engines such as Spark Streaming or Storm.

Apache Kafka is a distributed publisher-subscriber messaging system that is built to be scalable and fast. It is composed of topics, or message brokers, where producers write data and consumers read data from these topics. Figure 7-10 shows the data flow from the smart objects

(producers), through a topic in Kafka, to the real-time processing engine. Due to the distributed nature of Kafka, it can run in a clustered configuration that can handle many producers and consumers simultaneously and exchanges information between nodes, allowing topics to be distributed over multiple nodes. The goal of Kafka is to provide a simple way to connect to data sources and allow consumers to connect to that data in the way they would like. The following sections describe several of these packages and discusses how they are used to collect or process data

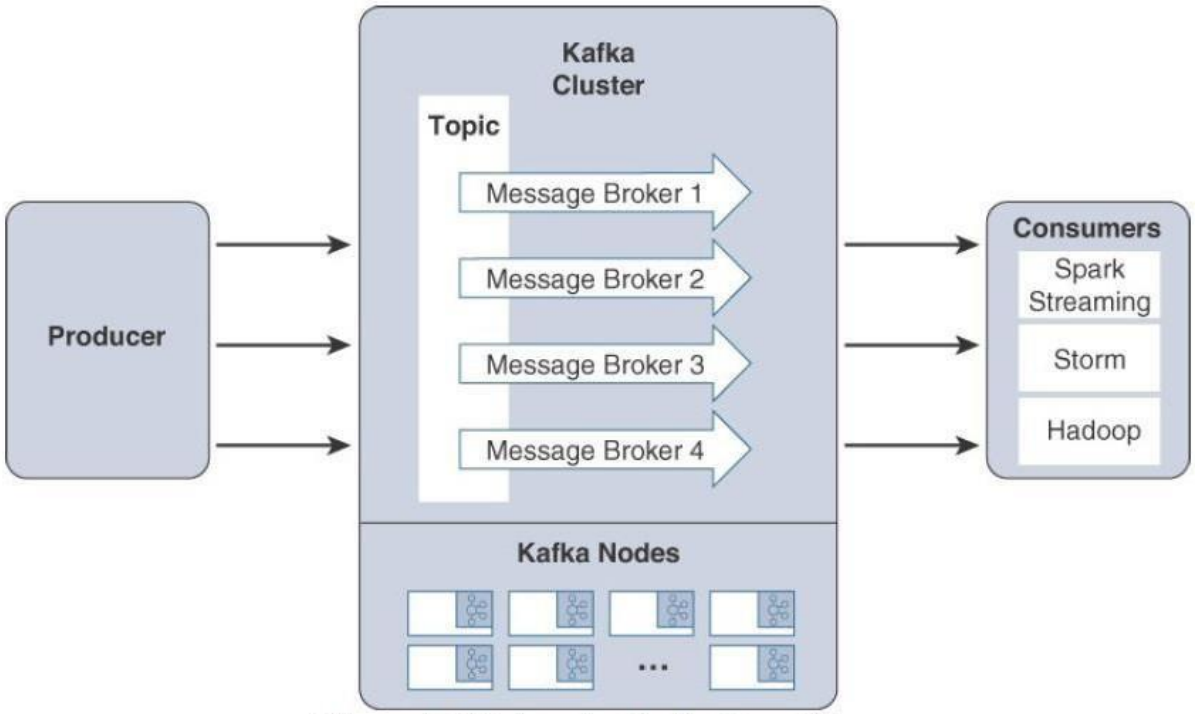


Figure 7-10 *Apache Kafka Data Flow*

Apache Spark

Apache Spark is an in-memory distributed data analytics platform designed to accelerate processes in the Hadoop ecosystem. The "in-memory" characteristic of Spark is what enables it to run jobs very quickly. At each stage of a MapReduce operation, the data is read and written back to the disk, which means latency is introduced through each disk operation. However, with Spark, the processing of this data is moved into high-speed memory, which has significantly lower latency. This speeds the batch processing jobs and also allows for near-real-time processing of events.

Apache Storm and Apache Flink

As you work with the Hadoop ecosystem, you will inevitably notice that different projects are very similar and often have significant overlap with other projects. This is the case with data streaming capabilities. For example, Apache Spark is often used for both distributed streaming analytics and batch processing. Apache Storm and Apache Flink are other Hadoop ecosystem projects designed

for distributed stream processing and are commonly deployed for IoT use cases. Storm can pull data from Kafka and process it in a near-real-time fashion, and so can Apache Flink. This space is rapidly evolving, and projects will continue to gain and lose popularity as they evolve.

Lambda Architecture

Ultimately the key elements of a data infrastructure to support many IoT use cases involves the collection, processing, and storage of data using multiple technologies. Querying both data in motion (streaming) and data at rest (batch processing) requires a combination of the Hadoop ecosystem projects discussed.

One architecture that is currently being leveraged for this functionality is the Lambda Architecture. Lambda is a data management system that consists of two layers for ingesting data (Batch and Stream) and one layer for providing the combined data (Serving). These layers allow for the packages discussed previously, like Spark and MapReduce, to operate on the data independently, focusing on the key attributes for which they are designed and optimized. Data is taken from a message broker, commonly Kafka, and processed by each layer in parallel, and the resulting data is delivered to a data store where additional processing or queries can be run. Figure 7-11 shows this parallel data flow through the Lambda Architecture.

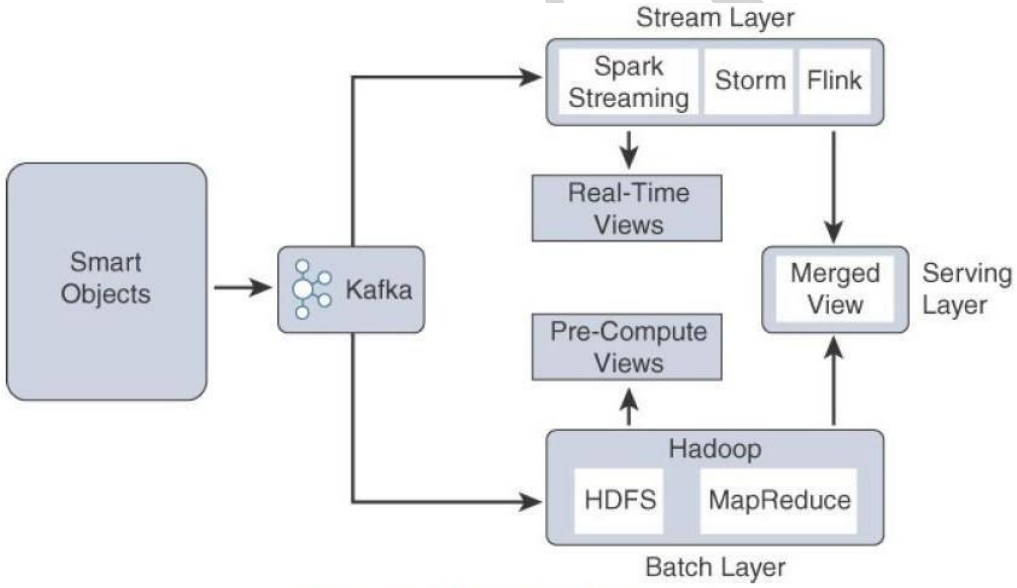


Figure 7-11 *Lambda Architecture*

The Lambda Architecture is not limited to the packages in the Hadoop ecosystem, but due to its breadth and flexibility, many of the packages in the ecosystem fill the requirements of each layer nicely:

- **Stream layer:** This layer is responsible for near-real-time processing of events. Technologies such as Spark Streaming, Storm, or Flink are used to quickly ingest, process, and analyze data on this layer. Alerting and automated actions can be triggered on events that require rapid response or could result in catastrophic outcomes if not handled immediately.
- **Batch layer:** The Batch layer consists of a batch-processing engine and data store. If an organization is using other parts of the Hadoop ecosystem for the other layers, MapReduce and HDFS can easily fit the bill. Other database technologies, such as MPPs, NoSQL, or data warehouses, can also provide what is needed by this layer.
- **Serving layer:** The Serving layer is a data store and mediator that decides which of the ingest layers to query based on the expected result or view into the data. If an aggregate or historical view is requested, it may invoke the Batch layer. If real-time analytics is needed, it may invoke the Stream layer. The Serving layer is often used by the data consumers to access both layers simultaneously.

4.3 Edge Streaming Analytics

One industry where data analytics is used extensively is the world of automobile racing. For example, in Formula One racing, each car has between 150 to 200 sensors that, combined, generate more than 1000 data points per second, resulting in hundreds of gigabytes of raw data per race. The sensor data is transmitted from the car and picked up by track-side wireless sensors. During a race, weather conditions may vary, tire conditions change, and accidents or other racing incidents almost always require an adaptable and flexible racing strategy. As the race develops, decisions such as when to pit, what tires to use, when to pass, and when to slow down all need to be made in seconds. Teams have found that enormous insights leading to better race results can be gained by analyzing data on the fly—and the data may come from many different sources, including trackside sensors, car telemetry, and weather reports.

Comparing Big Data and Edge Analytics

From a business perspective, streaming analytics involves acting on data that is generated while it is still valuable, before it becomes stale. For example, roadway sensors combined with GPS way finding apps may tell a driver to avoid a certain highway due to traffic. This data is valuable for only a small window of time. Historically, it may be interesting to see how many traffic accidents or blockages have occurred on a certain segment of highway or to predict congestion based on past traffic data. However, for the driver in traffic receiving this information, if the data is not acted upon immediately, the data has little value.

From a security perspective, having instantaneous access to analyzed and preprocessed data at the edge also allows an organization to realize anomalies in its network so those anomalies can be quickly contained before spreading to the rest of the network.

To summarize, the key values of edge streaming analytics include the following:

- **Reducing data at the edge:** The aggregate data generated by IoT devices is generally in proportion to the number of devices. The scale of these devices is likely to be huge, and so is the quantity of data they generate. Passing all this data to the cloud is inefficient and is unnecessarily expensive in terms of bandwidth and network infrastructure.

Analysis and response at the edge: Some data is useful only at the edge (such as a factory control feedback system). In cases such as this, the data is best analyzed and acted upon where it is generated.

- **Time sensitivity:** When timely response to data is required, passing data to the cloud for future processing results in unacceptable latency. Edge analytics allows immediate responses to changing conditions.

Edge Analytics Core Functions

To perform analytics at the edge, data needs to be viewed as real-time flows. Whereas big data analytics is focused on large quantities of data at rest, edge analytics continually processes streaming flows of data in motion. Streaming analytics at the edge can be broken down into three simple stages:

- **Raw input data:** This is the raw data coming from the sensors into the analytics processing unit.
- **Analytics processing unit (APU):** The APU filters and combines data streams (or separates the streams, as necessary), organizes them by time windows, and performs various analytical functions. It is at this point that the results may be acted on by micro services running in the APU.
- **Output streams:** The data that is output is organized into insightful streams and is used to influence the behavior of smart objects, and passed on for storage and further processing in the cloud. Communication with the cloud often happens through a standard publisher/subscriber messaging protocol, such as MQTT.

Distributed Analytics Systems

Depending on the application and network architecture, analytics can happen at any point throughout the IoT system. Streaming analytics may be performed directly at the edge, in the fog, or in the cloud data center. There are no hardand-fast rules dictating where analytics should be done, but there are a few guiding principles. We have already discussed the value of reducing the data at the edge, as well as the value of analyzing information so it can be responded to before it gets stale. There is also value in stepping back from the edge to gain a wider view with more data. It's hard to see the forest when you are standing in the middle of it staring at a tree. In other words, sometimes better insights can be gained and data responded to more intelligently when we step back from the edge and look at a wider data set.

Figure 7-15 shows an example of an oil drilling company that is measuring both pressure and temperature on an oil rig. While there may be some value in doing analytics directly on the edge, in this example, the sensors communicate via MQTT through a message broker to the fog analytics node, allowing a broader data set.

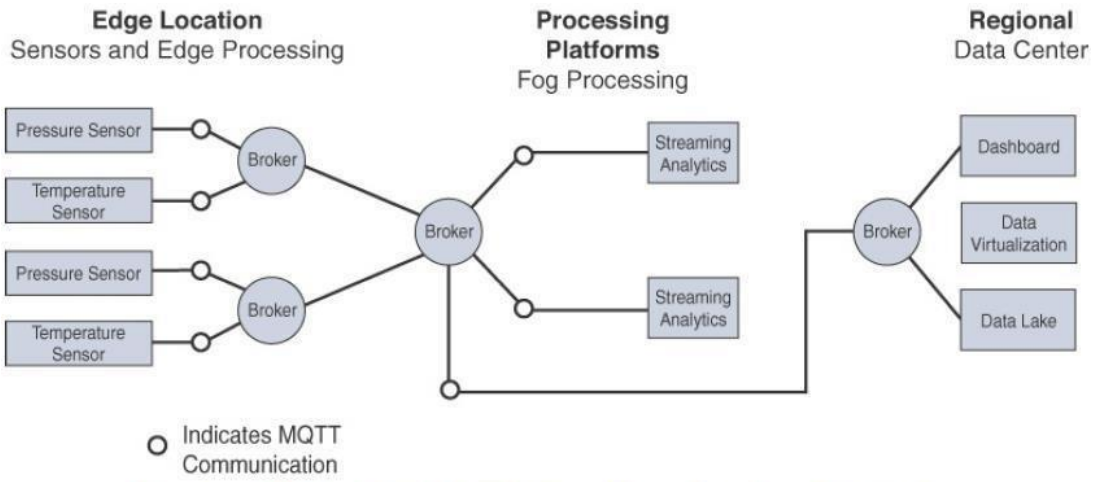


Figure 7-15 *Distributed Analytics Throughout the IoT System*

Network Analytics

Network analytics has the power to analyze details of communications patterns made by protocols and correlate this across the network. It allows you to understand what should be considered normal behavior in a network and to quickly identify anomalies that suggest network problems due to suboptimal paths, intrusive malware, or excessive congestion. Analysis of traffic patterns is one of the most powerful tools in an IoT network engineer's troubleshooting arsenal.

This behavior represents a key aspect that can be leveraged when performing network analytics: Network analytics offer capabilities to cope with capacity planning for scalable IoT deployment as well as security monitoring in order to detect abnormal traffic volume and patterns (such as an unusual traffic spike for a normally quiet protocol) for both centralized or distributed architectures, such as fog computing.

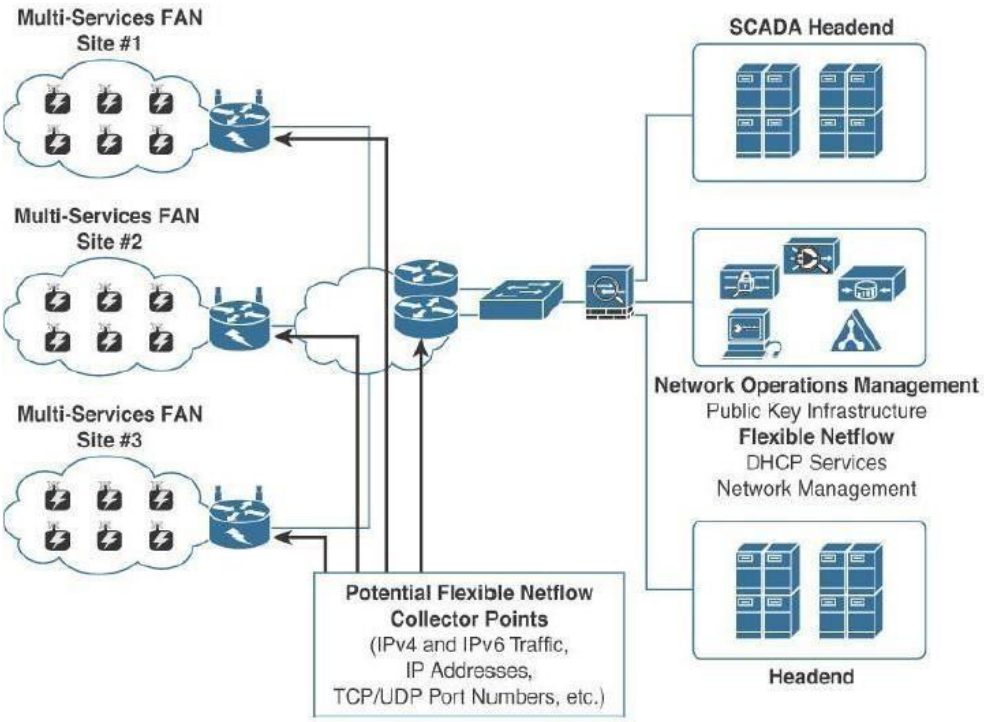


Figure 7-16 *Smart Grid FAN Analytics with NetFlow Example*

Consider that an IoT device sends its traffic to specific servers, either directly to an application or an IoT broker with the data payload encapsulated in a given protocol. This represents a pair of source and destination addresses, as well as application layer–dependent TCP or UDP port numbers, which can be used for network analytics.

4.4 Securing IoT

This chapter provides a historical perspective of OT security, how it has evolved, and some of the common challenges it faces. It also details some of the key differences between securing IT and OT environments. Finally, it explores a number of practical steps for creating a more secure industrial environment, including best practices in introducing modern IT network security into legacy industrial environments.

Common Challenges in OT Security

The security challenges faced in IoT are by no means new and are not limited to specific industrial environments. The following sections discuss some of the common challenges faced in IoT.

Erosion of Network Architecture

There is a wide variety in secured network designs within and across different industries. For

example, power utilities have a strong history of leveraging modern technologies for operational activities, and in North America there are regulatory requirements in place from regulatory authorities, such as North American Electric Reliability Corporation's (NERC's) Critical Infrastructure Protection (CIP)

Pervasive Legacy Systems

Due to the static nature and long lifecycles of equipment in industrial environments, many operational systems may be deemed legacy systems. For example, in a power utility environment, it is not uncommon to have racks of old mechanical equipment still operating alongside modern intelligent electronic devices (IEDs). In many cases, legacy components are not restricted to isolated network segments but have now been consolidated into the IT operational environment. From a security perspective, this is potentially dangerous as many devices may have historical vulnerabilities or weaknesses that have not been patched and updated, or it may be that patches are not even available due to the age of the equipment.

Insecure Operational Protocols

The structure and operation of most of these protocols is often publicly available. While they may have been originated by a private firm, for the sake of interoperability, they are typically published for others to implement. Thus, it becomes a relatively simple matter to compromise the protocols themselves and introduce malicious actors that may use them to compromise control systems for either reconnaissance or attack purposes that could lead to undesirable impacts in normal system operation.

Device Insecurity

Beyond the communications protocols that are used and the installation base of legacy systems, control and communication elements themselves have a history of vulnerabilities.

To understand the nature of the device insecurity, it is important to review the history of what vulnerabilities were discovered and what types of devices were affected. A review of the time period 2000 to 2010 reveals that the bulk of discoveries were at the higher levels of the operational network, including control systems trusted to operate plants, transmission systems, oil pipelines, or whatever critical function is in use.

4.5 How IT and OT Security Practices and Systems Vary

The differences between an enterprise IT environment and an industrial-focused OT deployment are important to understand because they have a direct impact on the security practice applied to them.

The Purdue Model for Control Hierarchy

Regardless of where a security threat arises, it must be consistently and unequivocally treated. IT information is typically used to make business decisions, such as those in process optimization, whereas OT information is instead characteristically leveraged to make physical decisions, such as closing a value, increasing pressure, and so on. Thus, the operational domain must also address physical safety and environmental factors as part of its security strategy —and this is not normally associated with the IT domain. Organizationally, IT and OT teams and tools have been historically separate, but this has begun to change, and they have started to converge, leading to more traditionally ITcentric solutions being introduced to support operational activities. For example, systems such as firewalls and intrusion prevention systems (IPS) are being used in IoT networks.

Figure 8-3 *The Logical Framework Based on the Purdue Model for Control Hierarchy*

This model identifies levels of operations and defines each level. The enterprise and operational domains are separated into different zones and kept in strict isolation via an industrial demilitarized zone (DMZ):

4.5.1 Enterprise zone

- **Level 5: Enterprise network:** Corporate-level applications such as Enterprise Resource Planning (ERP), Customer Relationship Management (CRM), document management, and services such as Internet access and VPN entry from the outside world exist at this level.
- **Level 4: Business planning and logistics network:** The IT services exist at this level and may include scheduling systems, material flow applications, optimization and planning systems, and local IT services such as phone, email, printing, and security monitoring.

4.5.2 Industrial demilitarized zone

- **DMZ:** The DMZ provides a buffer zone where services and data can be shared between the operational and enterprise zones. It also allows for easy segmentation of organizational control. By default, no traffic should traverse the DMZ; everything should originate from or terminate on this area.

4.5.3 Operational zone

- **Level 3: Operations and control:** This level includes the functions involved in managing the workflows to produce the desired end products and for monitoring and controlling the entire operational system. This could include production scheduling, reliability assurance, systemwide control optimization, security management, network management, and potentially other required IT services, such as DHCP, DNS, and timing
- **Level 2: Supervisory control:** This level includes zone control rooms, controller status, control system network/application administration, and other control-related applications, such as human-machine interface (HMI) and historian.
- **Level 1: Basic control:** At this level, controllers and IEDs, dedicated HMIs, and other applications may talk to each other to run part or all of the control function.
- **Level 0: Process:** This is where devices such as sensors and actuators and machines such as drives, motors, and robots communicate with controllers or IEDs.

4.5.4 Safety zone

- **Safety-critical:** This level includes devices, sensors, and other equipment used to manage the safety functions of the control system.

OT Network Characteristics Impacting Security

While IT and OT networks are beginning to converge, they still maintain many divergent characteristics in terms of how they operate and the traffic they handle. These differences influence how they are treated in the context of a security strategy. For example, compare the nature of how traffic flows across IT and OT networks:

4.5.5 **IT networks:** In an IT environment, there are many diverse data flows. The communication data flows that emanate from a typical IT endpoint travel relatively far. They frequently traverse the network through layers of switches and eventually make their way to a set of local or remote servers, which they may connect to directly.

4.5.6 **OT networks:** By comparison, in an OT environment (Levels 0–3), there are typically two types of operational traffic. The first is local traffic that may be contained within a specific package or area to provide local monitoring and closed-loop control. This is the traffic that is used for realtime (or near-real-time) processes and does not need to leave the process control levels.

Security Priorities: Integrity, Availability, and Confidentiality

In the IT business world, there are legal, regulatory, and commercial obligations to protect data, especially data of individuals who may or may not be employed by the organization. This emphasis on privacy focuses on the confidentiality, integrity, and availability of the data—not necessarily on a system or a physical asset. The impact of losing a compute device is considered minimal compared to the information that it could hold or provide access to. By way of comparison, in the OT world, losing a device due to a security vulnerability means production stops, and the company cannot perform its basic operation. Loss of information stored on these devices is a lower concern, but there are certainly confidential data sets in the operating environment that may have economic impacts, such as formulations and processes.

Security Focus

Security focus is frequently driven by the history of security impacts that an organization has experienced. In an IT environment, the most painful experiences have typically been intrusion campaigns in which critical data is extracted or corrupted. The result has been a significant investment in capital goods and human power to reduce these external threats and minimize potential internal malevolent actors. In the OT space, the history of loss due to external actors has not been as long, even though the potential for harm on a human scale is clearly significantly higher. The result is that the security events that have been experienced have come more from human error than external attacks. Interest and investment in industrial security have primarily been in the standard access control layers. Where OT has diverged, to some degree, is to emphasize the application layer control between the higher-level controller layer and the receiving operating layer. Later in this chapter you will learn more about the value and risks associated with this approach.

4.6 Formal Risk Analysis Structures: OCTAVE and FAIR

The key for any industrial environment is that it needs to address security holistically and not just focus on technology. It must include people and processes, and it should include all the vendor ecosystem components that make up a control system.

OCTAVE

OCTAVE (Operationally Critical Threat, Asset and Vulnerability Evaluation) has undergone multiple iterations. The version this section focuses on is OCTAVE Allegro, which is intended to be a lightweight and less burdensome process to implement. Allegro assumes that a robust security team is not on standby or immediately at the ready to initiate a comprehensive security review. This approach and the assumptions it makes are quite appropriate, given that many operational technology areas are similarly lacking in security-focused human assets. Figure 8-5 illustrates the OCTAVE Allegro steps and phases

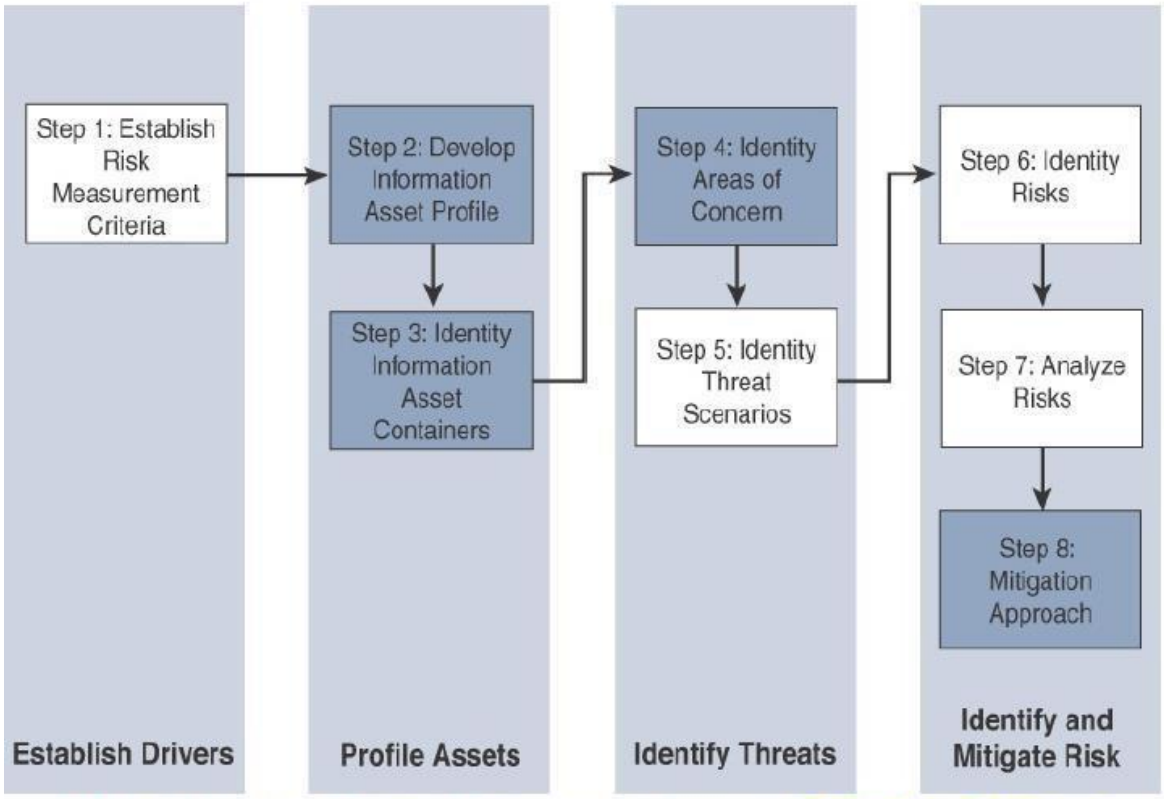


Figure 8-5 *OCTAVE Allegro Steps and Phases (see https://blog.compass-*

OCTAVE is a balanced information-focused process. What it offers in terms of discipline and largely unconstrained breadth, however, is offset by its lack of security specificity. There is an assumption that beyond these steps are seemingly means of identifying specific mitigations that can be mapped to the threats and risks exposed during the analysis process.

FAIR

FAIR (Factor Analysis of Information Risk) is a technical standard for risk definition from The Open Group. While information security is the focus, much as it is for OCTAVE, FAIR has clear applications within operational technology. Like OCTAVE, it also allows for non-malicious actors as a potential cause for harm, but it goes to greater lengths to emphasize the point. For many operational groups, it is a welcome acknowledgement of existing contingency planning. Unlike with OCTAVE, there is a significant emphasis on naming, with risk taxonomy definition as a very specific target.

FAIR places emphasis on both unambiguous definitions and the idea that risk and associated attributes are measurable. Measurable, quantifiable metrics are a key area of emphasis, which should lend itself well to an operational world with a richness of operational data. At its base, FAIR has a definition of risk as the probable frequency and probable magnitude of loss. With

this definition, a clear hierarchy of sub-elements emerges, with one side of the taxonomy focused on frequency and the other on magnitude.

Loss even frequency is the result of a threat agent acting on an asset with a resulting loss to the organization. This happens with a given frequency called the threat event frequency (TEF), in which a specified time window becomes a probability. There are multiple sub-attributes that define frequency of events, all of which can be understood with some form of measurable metric. Threat event frequencies are applied to a vulnerability. *Vulnerability* here is not necessarily some compute asset weakness, but is more broadly defined as the probability that the targeted asset will fail as a result of the actions applied. There are further sub-attributes here as well.

4.7 The Phased Application of Security in an Operational Environment

It is a security practitioner's goal to safely secure the environment for which he or she is responsible. For an operational technologist, this process is different because the priorities and assets to be protected are highly differentiated from the better-known IT environment.

Secured Network Infrastructure and Assets

Given that networks, compute, or operational elements in a typical IoT or industrial system have likely been in place for many years and given that the physical layout largely defines the operational process, this phased approach to introducing modern network security begins with very modest, non-intrusive steps.

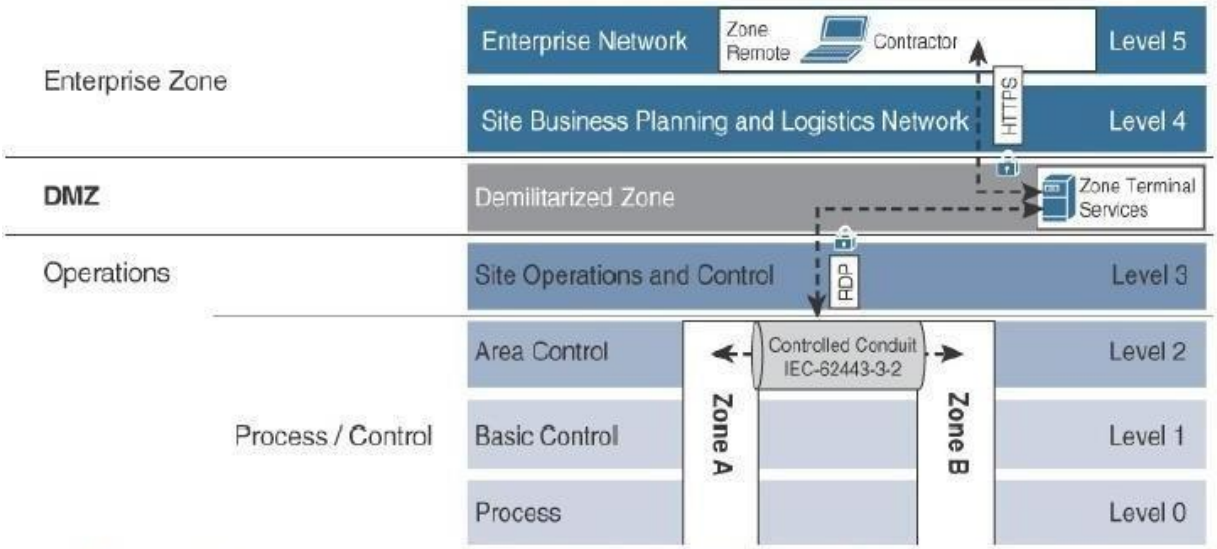


Figure 8-6 *Security Between Levels and Zones in the Process Control Hierarchy Model*

Normal network discovery processes can be highly problematic for older networking equipment. In fact, the discovery process in pursuit of improved safety, security, and operational state can result in degradation of all three.

Deploying Dedicated Security Appliances

The next stage is to expand the security footprint with focused security functionality. The goal is to provide visibility, safety, and security for traffic within the network. Visibility provides an understanding of application and communication behavior. With visibility, you can set policy actions that reflect the desired behaviors for inter-zone and conduit security. While network elements can provide simplified views with connection histories or some kind of flow data, you get a true understanding when you look within the packets on the network. This level of visibility is typically achieved with deep packet inspection (DPI) technologies such as intrusion detection/prevention systems (IDS/IPS). These technologies can be used to detect many kinds of traffic of interest, from simply identifying what applications are speaking, to whether communications are being obfuscated, to whether exploits are targeting vulnerabilities, to passively identifying assets on the network.

With the goal of identifying assets, an IDS/IPS can detect what kind of assets are present on the network. Passive OS identification programs can capture patterns that expose the base operating systems and other applications communicating on the network. The organizationally unique identifier (OUI) in a captured MAC address, which could have come from ARP table exploration, is yet another means of exposure. Coupled with the physical and historical data mentioned before, this is a valuable tool to expand on the asset inventory without having to dangerously or intrusively prod critical systems.

Higher-Order Policy Convergence and Network Monitoring

Another security practice that adds value to a networked industrial space is convergence, which is the adoption and integration of security across operational boundaries. This means coordinating security on both the IT and OT sides of the organization. Convergence of the IT and OT spaces is merging, or at least there is active coordination across formerly distinct IT and OT boundaries. From a security perspective, the value follows the argument that most new networking and compute technologies coming to the operations space were previously found and established in the IT space. It is expected to also be true that the practices and tools associated with those new technologies are likely to be more mature in the IT space.

There are advanced enterprise-wide practices related to access control, threat detection, and many other security mechanisms that could benefit OT security.

As stated earlier, the key is to adjust the approach to fit the target environment. Several areas are more likely to require some kind of coordination across IT and OT environments. Twosuch areas are remote access and threat detection. For remote access, most large industrial organizations backhaul communication through the IT network. Some communications, such as email and web browsing, are obvious communication types that are likely to touch shared IT infrastructure. Often vendors or consultants who require some kind of remote access to OT assets also traverse the IT side of the

network. Given this, it would be of significant value for an OT security practitioner to coordinate access control policies from the remote initiator across the Internet-facing security layers, through the core network, and to a handoff point at the industrial demarcation and deeper, toward the IoT assets.

The use of common access controls and operational conditions eases and protects network assets to a greater degree than having divergent groups creating ad hoc methods. Using location information, participant device security stance, user identity, and access target attributes are all standard functions that modern access policy tools can make use of. Such sophistication is a relatively new practice in industrial environments, and so, if these functions are available, an OT security practitioner would benefit from coordination with his or her IT equivalents.

Module- 5
IoT Physical Devices and Endpoints

Introduction to ARDUINO

- Arduino is an open-source advancement prototyping (development model) platform which depends on simple to utilize equipment and programming.
- Instructions to the microcontroller are given by the use of Arduino programming.
- Arduino software(IDE-Integrated improvement environment)
- The Arduino is a small computer that you can program to read information from the world around you and to send commands to the outside world.
- Arduino is a tiny computer that you can connect to electrical circuits.
- The brain of this board (Arduino Uno) is an ATmega328p chip (Micro controller) where you can store your programs that will tell your Arduino what to do.

5.1) Why Arduino?

1) Arduino is a open source product , software/hardware which is accessible and flexible to customers.
2) Arduino is flexible because of offering variety of digital and analog pins , SPI (Serial Peripheral Interface) and PWM
3) The **PWM pins** are used for giving the desired analog output .They are used to set the LED brightness or to run Stepper or Servo Motor or anything which require analog inputs 3, 5, 6, 9, 11 (PWM)
4) Serial Peripheral Interface (SPI) :- is an interface bus commonly used to send data between microcontrollers and small peripherals such as sensors, and SD cards.
5) Arduino is easy to use, connected to a computer via a USB and communicates using serial protocol .
6) Inexpensive around 500 rupees per board with free authoring software.
7) Ardunio has growing online community where lots of source code is available for use .
8) Ardunio is Cross-platform, which can work on windows,Mac or Linux platforms.
9) Ardunio follows simple, clear programming environment as C-language.

5.1.2) Which Arduino?

- In the ten years since Arduino was released , hundreds of “Arduino boards” are available in the market serving every kind of purpose.
- We focus on Arduino UNO .
- Some of the Boards from Arduino family are given below.

- Arduino mega is a big sister to the UNO with more memory and pins with a different chip the Atmega2560,useful when your project doesn"t fits in an UNO.
- Arduino Micro is bit smaller with a chip Atmega32u4 that can act like a keyboard or mouse which does its task with native USB.
- Its slim with downward pins which can be plugged into a breadboard.
- The Arduino MKR1000 is a little like an Arduino Micro but has a more powerful 32-bit ATSAM ARM chip and built-in WiFi.
- A great upgrade for when you want to do internet of Things projects.
- Flora is an Arduino compatible from Adafruit which is a round wearable which can be sewed (attach) into clothes.

5.2) Exploring ARDUINO UNO Learning Board.

- ❖ Microcontroller:- The ATmega328p is the Arduino brain.Everything on the Arduino board is meant to support this microcontroller
- ❖ Digital pins- 0-13
- ❖ For input or output
  - Apply 5v (HIGH)
  - 0V (LOW)

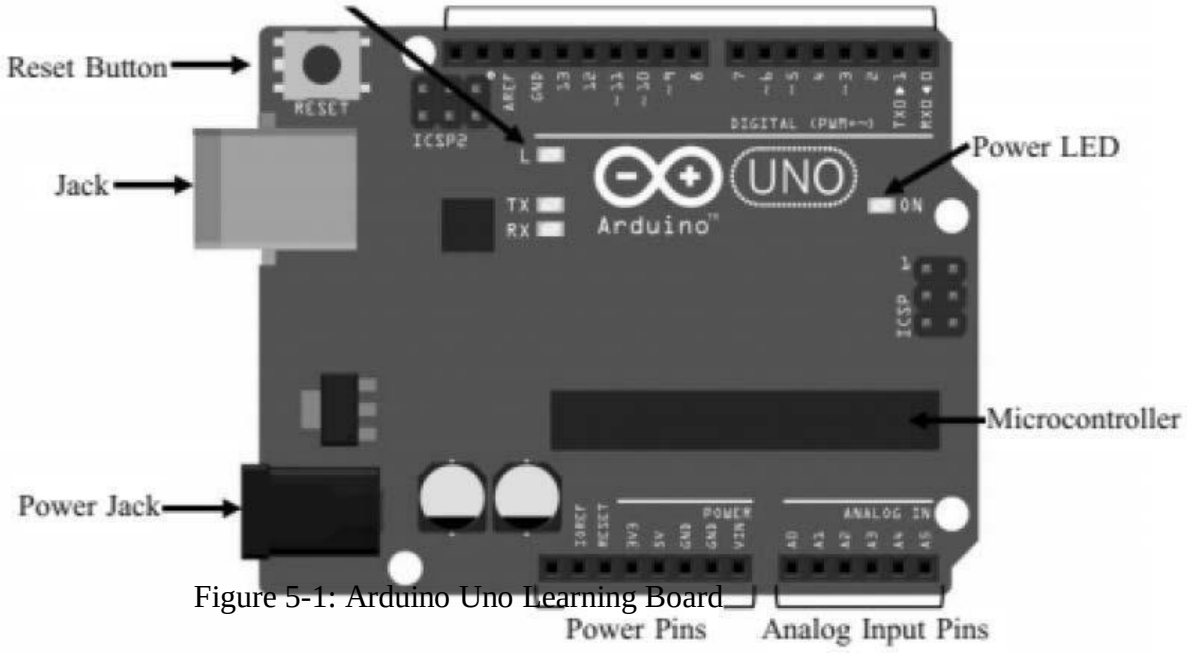


Figure 5-1: Arduino Uno Learning Board

- PWM pins: These are digital pins marked with a ~ (pins 11, 10, 9, 6, 5 and 3). PWM stands for "pulse width modulation" and allows to make digital pins output "fake" varying amounts of voltage. You'll learn more about PWM later.
- TX and RX pins: digital pins 0 and 1. The T stands for "transmit" and the R for "receive". Arduino uses these pins to communicate with the computer. Avoid using these pins, unless you're running out of pins.
- LED attached to digital pin 13: This is useful for an easy debugging of the Arduino sketches.
- TX and RX pins: these pins blink when there are information being sent between the computer and the Arduino.
- Analog pins: the analog pins are labeled from A0 to A5 and are most often used to read analog sensors. They can read different amounts of voltage between 0 and 5V. Additionally, they can also be used as digital output/input pins like the digital pins.
- Power pins: The Arduino has 3.3V or 5V supply, which is really useful since most components require 3.3V or 5V. The pins labelled as "GND" are the ground pins.
- Reset button: when you press that button, the program that is currently being run in your Arduino will start from the beginning. You also have a Reset pin next to the power pins that acts as reset button. When you apply a small voltage to that pin, it will reset the Arduino.
- Power ON LED: will be on since power is applied to the Arduino.
- USB jack: Connecting a male USB A to male USB B cable is how you upload programs from your computer to your Arduino board. This also powers your Arduino.
- Power jack: The power jack is where you connect a component to power up your Arduino (recommended voltage is 5V). There are several ways to power up your Arduino: rechargeable batteries, disposable batteries, wall-warts and solar panel.

5

- The simplest thing you can control with your Arduino is an LED.
- You can also display a message in a display, like the LCD display.
- You can also control DC or servo motors.
- You can also Read data from the outside world
- Motion sensor: The motion sensor allows you detect movement.
- Light sensor: this allows you to "measure" the quantity of light in the outside world.
- Humidity and temperature sensor: this is used to measure the humidity and temperature.
- Ultrasonic sensor: this sensor allows to determine the distance to an object through sonar.
- Shields – an extension of the Arduino
- Shields are boards that will expand the functionalities of your Arduino. You just need to plug them over the top of the Arduino. There are countless types of shields to do countless tasks.

- You can load new programs onto the main chip, the ATmega328p,via USB using the Arduino IDE.
- Browse on the following link:.Select which Operating System you are using and download.

- https://www.arduino.cc/en/Main/Software

Figure 5.2 : Arduino IDE

5.3.1) Connecting Arduino Uno Learning Board.

- ❖ After connecting your Arduino with a USB cable, you need to make sure that Arduino IDE has selected the right board you are using.
- ❖ In our case , we are using Arduino Uno,so you should go to Tools>Board"Arduino/Genuino Uno"> Arduino/Genuino Uno as shown in figure 5- 3.
- ❖ Then you select the right serial port where your Arduino is connected to.
- ❖ Go to Tools>port and select the right port as shown in Figure 5-4 and Figure 5-5shows the layout of ArduinoIDE.

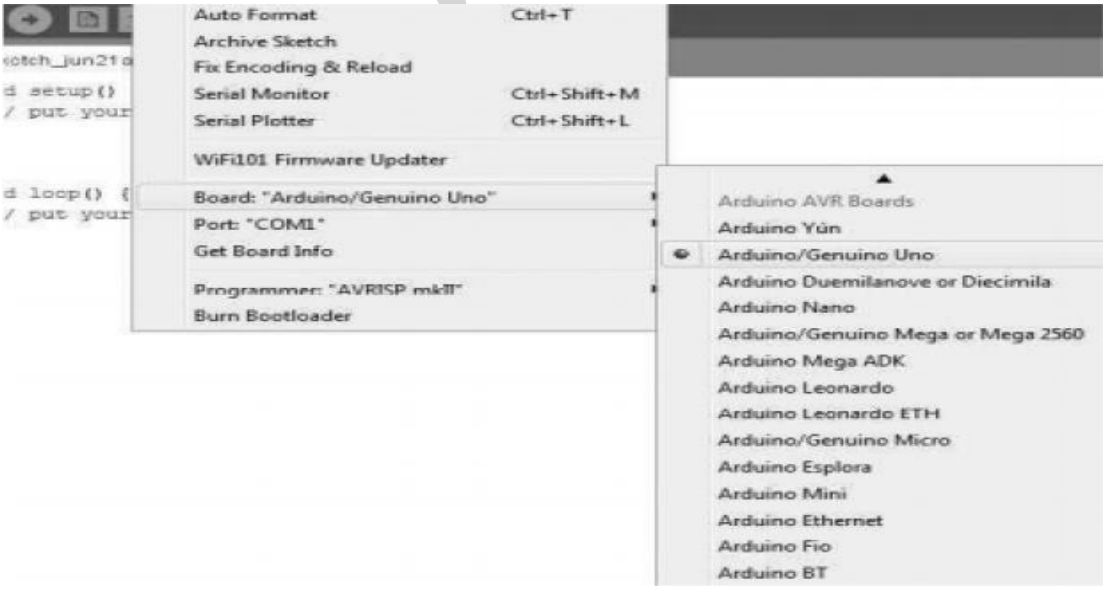


Figure 5.3 : Selecting the right Board

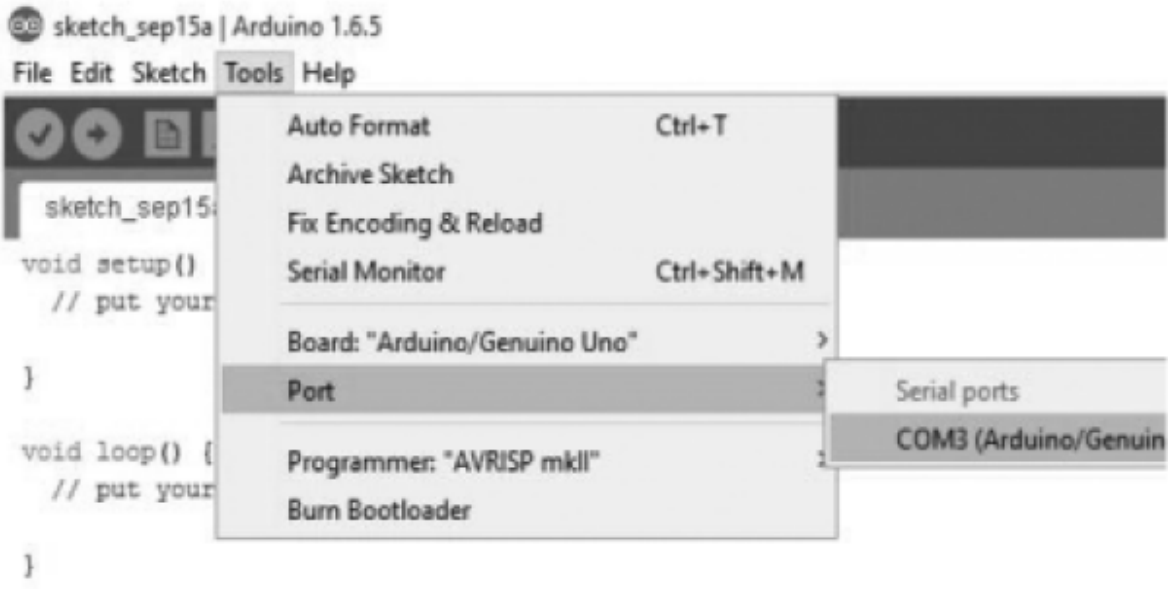


Figure 5.4 : Selecting the right port

Figure 5.5 : Layout of Arduino Uno IDE.

Breadboard for prototyping Arduino Uno Circuits

- In order to keep your circuit organized you need to use a breadboard, pictured below in Figure 5-6.
- The breadboard allows you to connect components together by plugging them into the little holes.

Z

- The key is to understand how the holes are connected.

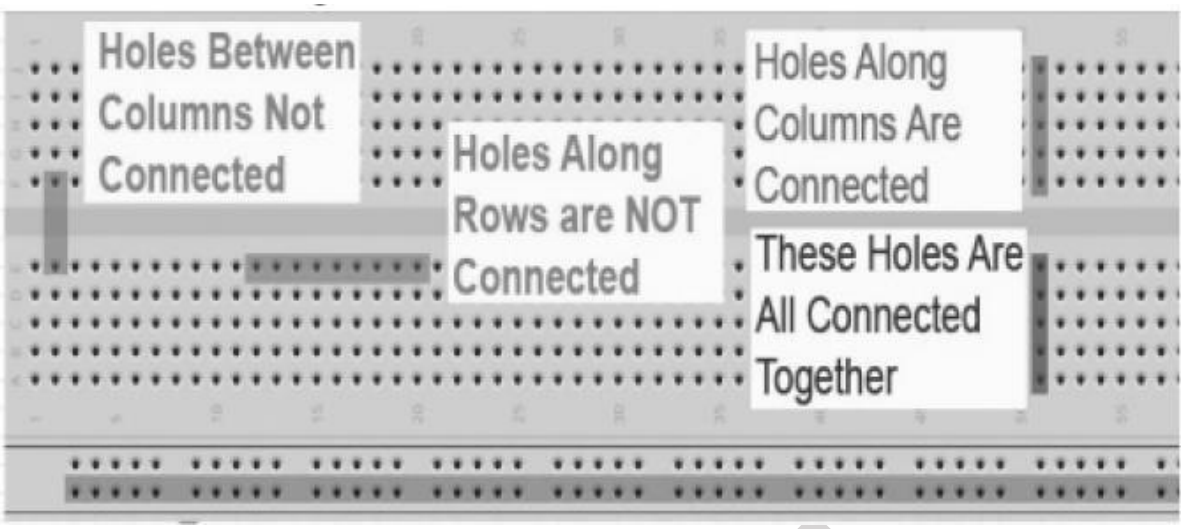


Figure 5.6 : Breadboard for prototyping Arduino Uno circuits.

Technical Specifications: Microcontroller Arduino UNO	A Tmega 328P
Operating Voltage	5V
Input Voltage (recommended)	7–12V
Input Voltage (limit)	6–20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pin	6
Analog Input Pins	6
DC Current per I/O Pin	20mA
DC Current for 3.3V Pin	50mA
Flash Memory	32 KB (A Tmega 328P) of which 0.5 KB used by bootloader
SRAM	2KB (A Tmega328P)
EEPROM	1 KB (A Tmega 328P)
Clock Speed	16MHz

Table-1 Technical Specification of Arduino UNO

5.4) Fundamentals of Arduino Programming

In this section explains the basic structure of Arduino programming with respect to usage of variables, constants , control flow statement and finally the predefined functions used to read analog and digital inputs.

Structure	The structure of Arduino programming contains of two parts as shown below
	void setup() //Preparation function used to declare variables
	{ //First function that runs only one in the program
	Statement(s); //used to set pins for serial communication
	}
	void loop() //Execution block where instructions are executed repeatedly
	{ //this is the core of the Arduino programming
	Statements(); //Functionalities involve reading inputs, triggering outputs etc.
	}

void setup()	void setup()
	{
	pinMode(pin, INPUT); //'pin' configure as inpu
	}

void loop()	void loop() //After calling setup(),loop() function does its task
	{
	digitalWrite(pin, HIGH); //sets 'pin' ON
	delay(10000); //pauses for ten thousand mili second
	digitalWrite(pin, LOW); //sets 'pin' OFF
	delay(10000); //pauses for ten thousand mili second
	}

Functions	A function is a piece of code that has a name and set of statements executed when function is called. Functions are declared by its type followed with name of a function. Syntax: type functionName(parameters)

	{
	Statement(s);
	}
	Example:
	int delayvar()
	{
	int var; //create temporary variable var
	var=analogRead(potent); //read from potentiometer
	var=var/4; //convert the value of variable var
	return var; //return var
	}

{ } curly braces	They define beginning and end of function blocks, unbalanced braces may lead to compile errors.
semicolon	It is used to end a statement and separate elements of a program. Syntax: int x=14;
/*…..*/ block comments	Multiline comments begin with /* with a description of the block and ends with */. Syntax: /*This is an enclosed block of comments Use the closing comment to avoid errors*/
//line comments	Single line comment begins with // and ends with next instruction followed. Syntax: //This is a Single line comment

Variables	A variable is a way of storing value for later use in the program. A variable is defining by its value type as an int, long, float etc by setting a specified name and optionally assigning an initial value. A global variable can be seen in every part of the program which is declared at the beginning of program before setup() function. A local variable is defined inside a function in which it was declared.
	Example:
	int var; //variable 'var' visible to all functions
	void setup()
	{
	//nothing is required
	}
	void loop()

	{
	for(int local=0;local<5;)
	{
	local++;//variable 'local' is only visible within for loop
	}
	float local_f; //variable 'local_f' is visible only inside the loop
	}

Data Types	

Data type	Syntax	Range
Byte	byte x = 100;	0–255
Int	int y = 200;	32767 to–32768
Long	long var = 8000;	2147483647 to –2147483648
Float	float x = 3.14;	3.4028235E+38to – 3.4028235E+38
arrays	int myarray []={10,20,30,40}	Size depends on the data type associated with declaration.

<table>
<tr><td>Operators</td><td colspan="2"><table>
<tr><th>Operstor</th><th>Syntax and its usage</th></tr>
<tr><td>Arithmetic operators
(+,–,/,*)</td><td>x = x+5;
y–y–6;
z=z*2;
p=p/q;</td></tr>
<tr><td>Assignment operators
(=,++,—,+ =,=,*=,/=)</td><td>x++; //same as x=x+1
x+=y;//same asx=x+y
x–=y; //same asx=x–y
x*=y; //same as x=x*y
x/=y; //same as x=x/y</td></tr>
</table></td></tr>
<tr><td></td><td>Comparison operators
(==,!=,<>,<=,<=)</td><td>x==y //xis equal to y
x!–y //xis not equal to y
x<y //xis less that y
x!–y // x is not equal to y</td></tr>
<tr><td></td><td>Logical operators
(&&,||,!)</td><td>x>2&&x<5 //Evaluates to true only if both expression are true
x>2||y>2 //Evaluates to true if any one expression is true
!x>2 //true if only expression is false</td></tr>
</table>

Constants	Constants	Usage

	TRUE/FALSE	Boolean constants true=2 and false=0 defined in logic levels.
		if(b==TRUE)
		{
		//do something
		}

	INPUT/OUTPUT	Used with pinMode () function to define levels. pinMode (13,OUTPUT):
	HIGH/LOW	Used to define pin levels HIGH–1,ON,5 volts LOW –0,OFF, 0 volts Digital Write (13,HIGH);

Flow control Statements	

if	if(some_variable == value)
	{
	Statement(s); //Evaluated only if comparison results in a true value
	}

if…else	if(input==HIGH)
	{
	Statement(s); //Evaluated only if comparison results in a true value
	}
	else
	{
	Statement(s); //Evaluated only if comparison results in a false value
	}

for	for(initialization;condition;expression)
	{
	Dosomething; //Evaluated till condition becomes false
	}
	for(int p=0;p<5;p++) //declares p, tests if less than 5, increments by 1

	{
	digitalWrite(13,HIGH); //sets pin 13 ON
	delay(250); // pauses for ¼ second
	digitalWrite(13,LOW); //sets pin 13 OFF
	delay(250); //pause for ¼ second
	}

while	While loop executes until the expression inside parenthesis becomes false.
	while(some_variable ?? value)
	{
	Statement(s); //Evaluated till comparison results in a false value
	}

do...while	Bottom evaluated loop, works same way as while loop but condition is tested at the end of loop. do { Dosomething; }while(somevalue);

Digital and Analog input output pins and their usage		
Digital i/o	Methods	Usage

	pinMode (pin, mode)	Used in setup () method to configure pin to behave as INPUT/OUTPUT
		pinMode(pin, INPUT) //pin set to INPUT
		pinMode(pin, OUTPUT) // pin set to OUTPUT

	Digital Read (pin)	Read value from a specified pin with result being HIGH/LOW Val=digital Read(pin); // Val will be equal to input pin
	Digital Write(pin, value)	Outputs to HIGH/LOW on a specified pin. Digital Write(pin, HIGH); //pin is set to HIGH

	Example	int x=13; //connect 'x' to pin 13
		int p=7; //connect push button to pin 7
		int val=0; //variable to store the read value
		void setup()
		{
		pin MODE(x, OUTPUT); // sets 'x' as OUTPUT
		}
		void loop()
		{
		val=digital Read (p); //sets 'value' to 0
		digital Write (x,val); //sets 'x' to button value
		}

Analog i/o	**Methods**	**Usage**
	analog Read(pin)	Reads value from a specified analog pin works on pins 0–5. val=anolog Read (pin); // 'val' equal to pin
	analog Write (pin,value)	Writes an analog value using pulse width modulation (PWM) to a pin marked PWM works on pins 3,5,6,9,10.

	Example	int x=10; //connect 'x' to pin 13
		int p=0; //connect potentiometer to analog pin 7
		int val; //variable for reading
		void setup () { } //No setup is needed
		void loop()
		{
		val=analog Read (p); //sets 'value' to 0
		val/=4;
		analog Write (x,val); //outputs PWM signal to 'x'
		}

Z

time	Methods	Usage
	deaay (ms)	Pauses for amount of time specified in milliseconds. deay(1000); //waits for one second
	millis()	Returns the number of milliseconds since Arduino is running. val=millis();//'val' will be equal to millis()

math	Methods	Usage
	min(x,y)	Calculates minimum of two numbers val=min (val,10); //sets 'val' to smaller than 10 or equal to 10 but never gets above 10.
	max(x,y)	val=max (val, 10); //sets 'val' to larger than 100 or 100.
random	**Methods**	**Usage**
	Random Seed (value)	Sets a value / seed as starting point for ranction.
	Random (min, max)	Allows to return numbers within the range specified by min and max values. val=random(100,200); //sets 'val' to random number between 100-200

	Example	intr number ; //variable to store random value
		int x=10;
		void setup()
		{
		randomseed (millis()); //set millis() as seed
		number -random(200); //random number from 0–200
		analog Write(x, number); //outputs PWM signal
		delay (500);
		}

Serial	**Methods**	**Usage**

	Serial. begin(rate)	Opens serial port and sets the baud rate for serial data transmission.
		void setup()
		{
		Serial begin(9600); //sets default rate to 9600 bps
		}

	Serial. println (data)	Prints data to the serial port Serial println (value); //sends the 'value'; //sends the ' value ' to serial monitor.

5.4.1) Difference between Analog, Digital and PWM Pins

- In analog pins, you have unlimited possible states between 0 and 1023.
- This allows you to read sensor values.
- Example: With a light sensor, if it is very dark , you will read 1023,if it is very bright yow will read as 0.
- If the brightness between dark and very bright you will read a value between 0 and 1023.
- In digital pins you have just two possible states, which is on and off.
- These can also be referred as High or Low,1 or 0 and 5v or 0V
- If the LED is on then its state is High or 1 or 5 V. If it is off have Low 0 0r 0 v.
- PWM pins are digital pins , so they output either 0 or 5v
- However these pins can output fake intermediate voltage values between 0 and 5 v because they can perform " Pulse Code Modulation".

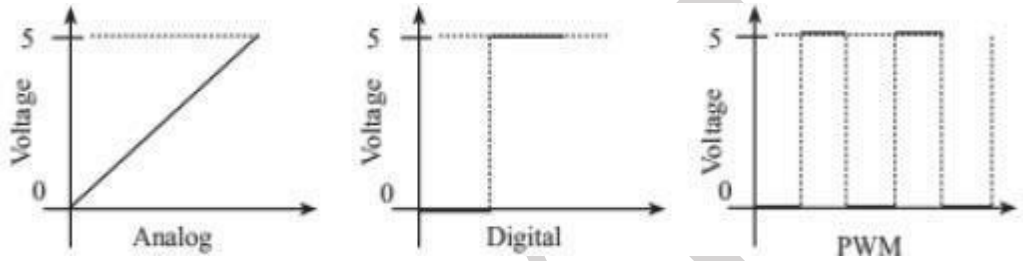


Figure 5.7 : Difference between Analog, Digital and PWM Pins.

5.5) Introduction to RaspberryPi

- The RaspberryPi is a series of credit card sized single-board computers developed in the United Kingdom by the Raspberrypi Foundation to promote the teaching of basic computer science in schools and developing countries.
- Several generations of RaspberryPi have been released.
- The first generation (RaspberryPi 1 model B) was released in February 2012, followed by a simple and inexpensive model A.
- In 2014 , the foundation released a board with an improved design in Raspberry 1 model B+.
- Improved A+ and B+ model were released a year later.
- RaspberryPi Zero with smaller size and limited input/output (I/O)and general purpose input/ output (GPIO) abilities was released in November 2015 for US $5.
- RaspberryPi 2 which added more RAM was released in February 2015.
- RaspberryPi 3 model B released in February 2016 in bundled with on-board Wi-Fi and Bluetooth.
- As of 2016 , Raspberrypi 3 model b is the newest mainline RaspberryPi. These boards are priced between US $ 5-35.

RaspberryPi	Model A+	Model B	Model B+	2, Model B	Model 3
Quick Summary	Cheapest,smallest single board computer	The original Raspberry Pi.	More USB and GPIIO than the B.Ideal choice for schools	most advanced Raspberry Pi.	Newest with wireless connectivity
Chip	Broadcom BCM 2835			Broadcom BCM2836	Broadcom BCM 2837
processor	ARMv6 single core			ARMv7 quad core	4×ARM Cortex-A53
Processor speed	700 MHz			900 MHz	1.2GHz
Voltage and power draw	600mA @ 5V			650mA @ 5V	
GPU	Dual core Videocore IV Multimedia Co-Processor				Broadcom Videocone IV
Size	65×56mm	85×56mm			
Memory	256 MB SDRAM @ 400 MHz	512 MB SDRAM @ 400 MHz		1 GB SDRAM @ 400 MHz	1 GB LPDDR2 (900 MHz)
Storage	Micro SD Card	SD Card	Micro SD Card	Micro SD Card	Micro SD Card
GPIO	40	26	40		
USB 2.0	1	2	4		
Ethernet	None	10/100mb Ethernet RJ45J ack			
Wireless	None				2.4GHz 802.11n wireless
Bluetooth	None				Bluetooth 4.1 Classic, Bluetooth Low Energy
Audio	Multi-Channel HD Audio over HDMI, Analog Stereo from 3.5mm Headphone Jack				

Operating Systems	Raspbian RaspBMC,Arch Linux,Rise OS,OpenELEC Pidora
Video Output	HDMI Composite RCA
Supported Resolutions	640×350 to 1920×1200, including 1080p,PAL & NTSC standards
Power Source	Micro USB

5.6) Exploring the RaspberryPi Learning Board

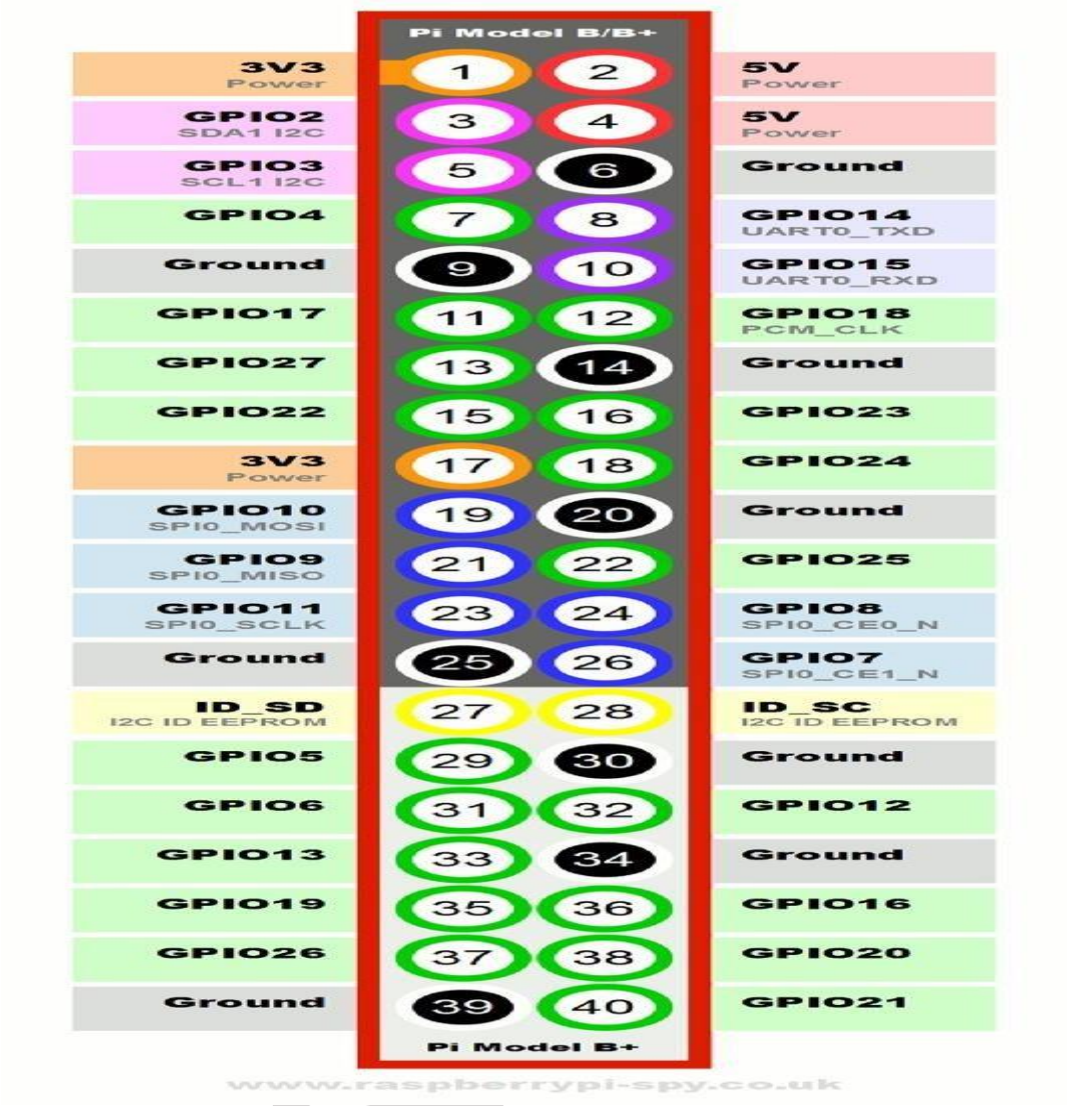


Figure 5.8: GPIO Pinout Diagram.

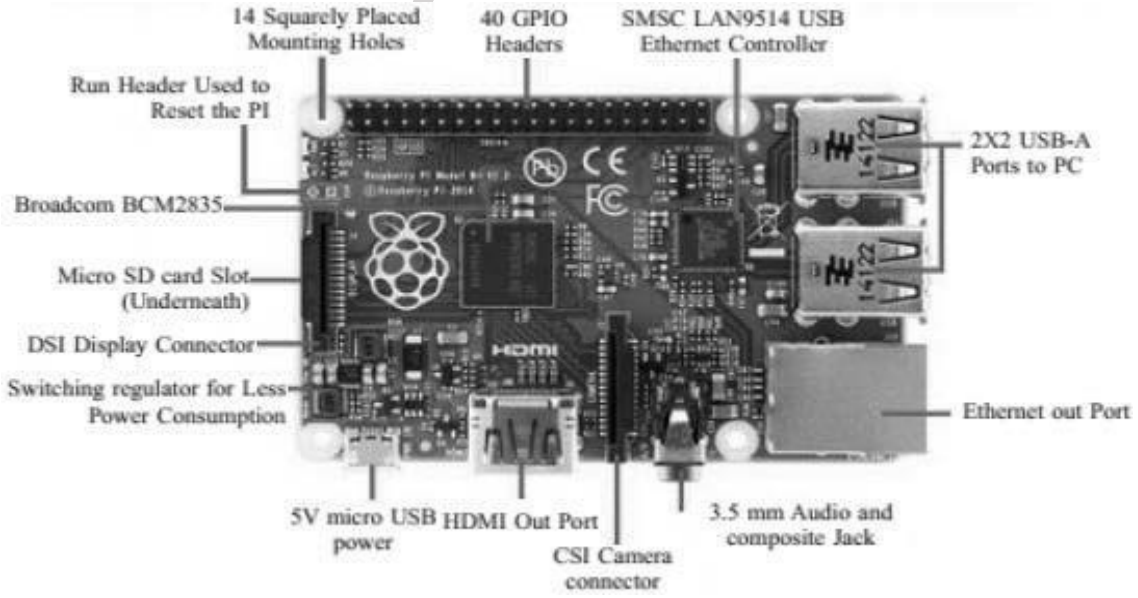


Figure 5.9: Raspberry Pi2 Model B and its GPIO.

❖ **Processor –** The Broadcom BCM2835 SoC(System on Chip) used in the first generation Raspberrypi is somewhat equivalent to the chip used in first generation smart phones, Which includes a 700MHz ARM1176JZF-S Processor, Video Core IV graphics processing unit(GPU) and RAM.

- This has a level 1 (L1) cache of 16 KB and a level 2 (L2) cache of 128 KB.
- The level 2 cache is used primarily by the GPU.
- The Raspberrypi2 uses a Broadcom BCM2836 SoC with a 900 MHz 32-bit quad-core ARM cortex A7 processor with 256KB shared L2 cache.
- The Raspberrypi2 uses a Broadcom BCM2837 SoC with a 1.2 GHz 64-bit quad-core ARM cortex A53 processor, with a 512 KB shared L2 cache.

❖ **Power Source**:- The recommended and easiest way to power the Raspberrypi is via the Micro USB port on the side of the unit.

- The recommended input voltage is 5V , and the recommended input current is 2A.

❖ **SD Card (Secure Digital Card**): The Raspberry Pi does not have any locally available storage accessible.

- The working framework is stacked on a SD card which is embedded on the SD card space on the Raspberry Pi.

❖ GPIO(General Purpose Input Output): General –purpose input/output (GPIO) is a non specific pins on a coordinated circuit to know is an input or output pin which can be controlled by the client at run time.

- GPIO capabilities may include

1. GPIO pins can be designed to be input or output.

2. Input values are meaning (normally high=1, low=0).

3. Yield values are writable/meaningful.

4. Input values can frequently be utilized as IRQs(interrupt request).

❖ **DSI Display x:** The Raspberrypi Connector S2 is a display serial interface(DSI) for connecting a liquid crystal display(LCD) panel using a 15 pin ribbon cable.

❖ **Audio Jack:** A standard 3.5 mm TRS connector is accessible on the Rpi for stereo sound yield.

- Any earphone or 3.5mm sound link can be associated straightforwardly.

❖ **Status LEDS:** There are 5 status LEDs on the Rpi that demonstate the status of different exercise

1. OK- SDCard Access.
2. POWER- 3.3 v Power
3. FDX- Full Duplex LAN
4. LNK- Link/ Activity(LAN) (Model B) 5.

10M/100-10/100M bit (LAN) (Model B)

- **Ethernet Port :**is accessible on Model B and B+.

- It can be associated with a system or web utilizing a standard LAN link on the Ethernet port.

- **CSI connector (CSI) –** Camera Serial interface is a serial interface outlined (define) by MIPI (Mobile Industry Processor Interface) organization together went for interfacing cameras with a portable processor.
- **HDMI-** High Definition Multimedia Interface to give both video and sound yield.

5.6.1) Description of system on chip(SoC)

- SoC is an integrated circuit(IC) that coordinated all parts of a PC or other electronic framework into a solitary chip.
- SoC comprises of:

1) A microprocessor chip or DSP core.

2) Memory pieces including (ROM,RAM,EEPROM)

3) Timing signal –oscillators

4) Peripherals include counter-clocks, ongoing clocks

5) Outer interfaces example: USB, Ethernet

6) Simple interfaces includes ADCs (Analog and Digital Converter) and DACS(Digital and Analog Converter)

7) Voltage Controllers and power administration circuits.

Accessories

- **Camera:** On 14 May 2013 , the establishment and the merchants RS Components and Premier Farnell/ Element 14 propelled the Raspberry pi camera board with a firmware redesign to bolster it.

- **Gertboard**- A Raspberry Pi Foundation authorized gadget intended for instructive purpose, and grows the Raspberry Pi's GPIO pins to permit interface with of LEDs , switches, sensors and different gadges.

5.7) Raspberry Pi interfaces.

- **Serial :-** The serial interface on Raspberry Pi has receive(rx) and transmit(Tx) pins for communication with serial peripherals.
- **SPI:-** Serial Peripheral interfaces(SPI) is a synchronous serial data protocol used for communication with one or more peripheral devices.
    - MISO (Master In Slave Out): Master line for sending data to the peripherals.
    - MOSI(Master out Slave In): Slave line for sending data to the master.
    - SCK(Serial Clock): Clock generated by Master to synchronize data transmission.
    - CEO(Chip Enable 0): To enable or disable device
    - CEO(Chip Enable 1): To enable or disable device
- **I2C:-** The I2C interface pins on Raspberry Pi allow you to connect hardware modules.

5.8)RaspberryPI Operating System.

- Various operating system can be installed on Raspberrypi through SD cards.

5.8.1) Operating Systems(not Linux based)

- ✓ RISC OS Pi (a special cut down version RISC OS Pico, for 16MB cards and larger for all models of Pi 1 and 2, has also been made available.
- ✓ FreeBSD
- ✓ NetBSD
- ✓ Plan 9 from Bell Labs and Inferno
- ✓ Windows 10 IoT core- a no cost edition of Windows 10 offered by Microsoft that runs natively on the RaspberryPi 2.
- ✓ xv6-is a modern reimplementation of sixth edition Unix OS for teaching purposes, it is ported to RaspberryPi from MIT Xv6, which can boot NOOBs.
- ✓ Haiku-is an open source BeOS clone that has can be compile for the RaspberryPi and several other ARM boards

5.8.2) Operating Systems(Linux based

- ✓ Xbian-using Kodi open source digital media center
- ✓ openSUSE
- ✓ Raspberry Pi Fedora remix
- ✓ Pidora, another Fedora Remix optimized for the RaspberryPi
- ✓ Gentoo Linux
- ✓ Diet Pi
- ✓ CentOS\OpenWrt
- ✓ Kali Linux
- ✓ Ark OS
- ✓ Kano OS
- ✓ Nard SDK

5.8.3) Media center Operating systems

- ✓ OSMC
- ✓ OpenELEC
- ✓ LibreELEC
- ✓ Xbian
- ✓ Rasplex

5.8.4) Audio Operating systems

- ✓ Volumio
- ✓ Pimusicbox
- ✓ Runeaudio
- ✓ moOdeaudio

5.8.5) Recalbox

- ✓ Happi Game Center
- ✓ Lakka
- ✓ ChameleonPi
- ✓ Piplay

5.9) Operating System Setup in RaspberryPI

- Preinstalled NOOBS operating system is already available in many authorized as well as independent seller, there are many other operating system for Raspberrypi in the market like NOOBS, Raspbian and third party operating systems are also available like UBUNTU MATE, OSMC,RISC OS etc.

Z

- To setup an operating system we need a SD card with minimum capacity of 8GB.

5.9.1) Formatting SD card

Format the SD card before copying NOOBS onto it. To do this

1. Download SD formatter 4.0 from SD Association website for either Windows or Mac.
2. Follow the instructions to install the software.
3. Insert the SD card into the computer or laptops SD card reader and make a note of the drive letter allocated to it
4. In SD formatter, select the drive letter the SD card is and format it.

5.9.2) OS installation

Follow the steps to install operating system in the SD card.

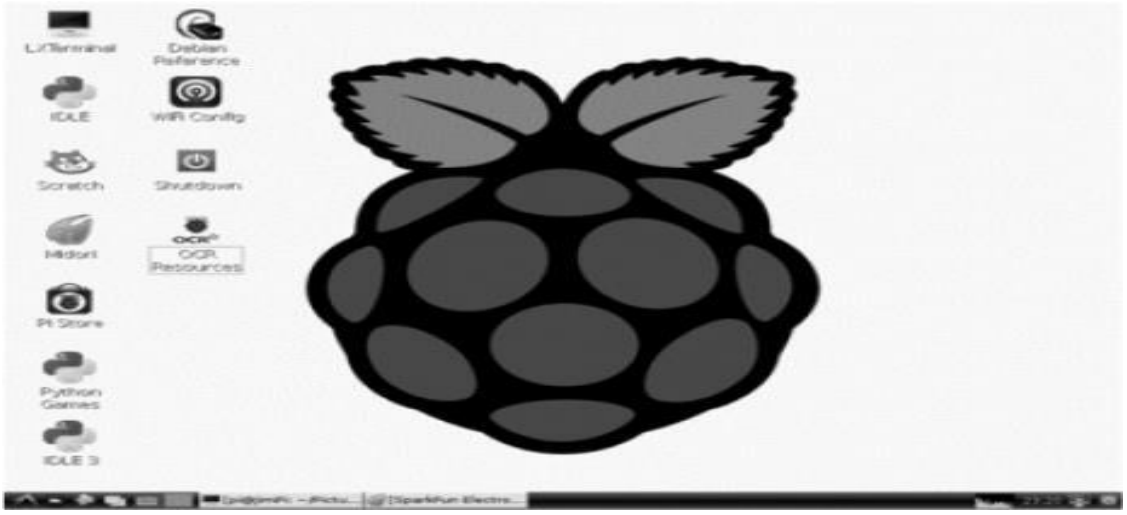
1. Go to raspberry pi foundation website and click on DOWNLOAD section.
2. Click on NOOBS, then click on the "Download ZIP" button under NOOBS(offline and network install) and select a folder to save this ZIP file
3. Extract all the files from ZIP.
4. Once SD card has been formatted, drag all the files in the extracted NOOBS folder and drop them onto the SD card drive.
5. The necessary files will then be transferred to the SD card.
6. When this process has finished, safely remove the SD card and insert it into the Raspberry Pi.

5.9.3) First Boot

1. Plug in the keyboard, mouse, and monitor cables.
2. Now plug the USB cable into the Raspberrypi.
3. Now Raspberrypi will boot, and a window will appear with a list of different operating systems that we can install. We recommend using Raspbian- tick the box next to Raspbian and clicking on install.
4. Raspbian will then run through its installation process. Note that this can take a while.
5. When the install process has completed, the Raspberrypi configuration menu (raspi-config) will load. Here we should set the time and date for our region, enable a Raspberrypi camera board, or even create users.

5.10) Login information

- The default login for Raspbian is username "pi" with the password "raspberry".
- To load the graphical user interface, type "Startx" and press Enter.



5.10) RaspberryPI Commands

General commands for RaspberryPi:

a. **raspi-config:** Configuration settings menu
b. **clear:** clears data from terminal.
c. **date:** current date.
d. **reboot:** Reboot immediately
e. **shutdown –h** now: Shutdown system immediately
f. **nano example.txt:** Opens the example.txt in the text editor nano.
g. **poweroff:** To shutdown immediately.
h. **shutdown –h** 01:22: To shutdown at 1:22 AM.
i. **apt-get update:** Synchronizes the list of packages on our system to the list in the repositories. Use this before installing new packages to make sure we are installing the latest version
j. **apt-get upgrade:** Upgrades all of the software packages we have installed.
k. **startx:** Opens the GUI

Directory and File commands:

a. **mkdir new_directory**: Creates a new directory named new_directory.
b. **mv new_folder:** Moves the file or directory named "new_folder" to a specified location
c. **rm new_file.txt:** Deleted the file new_file.txt.
d. **rmdir new_directory:** Deletes the directory "new_directory" only if it is empty.
e. **touch new_file.txt:** creates a new, empty file named new_file.txt in the current directory.
f. **cat new_file.txt:** Displays the contents of the file new_file.txt
g. **cd /xyz/abc:** Changes the current directory to the /xyz/abc directory.
h. **ls –l:** lists files in the directory.

Networking and Internet Commands:

a. **iwconfig:** Tp check which wireless adapter is currently active.
b. **ifconfig**: wireless connection ststus.
c. **ping:** tests connectivity between two devices connected on a network.
d. **wget http://www.website.com/new_file.txt**: Downloads the file new_file.txt from the web and saves it to the current directory.
e. **nmap:** Scans the network and lists connected devices, protocol, port number and other information.
f. **iwlist wlan0 scan:** list of currently available wireless networks.

System information commands:

a. **cat /proc/meminfo:** shows details about memory
b. **cat /proc/version:** shows which version of rsapberrypi we are using.
c. **df –h:** shows information about available disk space.
d. **df /:** shows how much free disk space is available.
e. **free:** shows how much free memory is available.
f. **hostname –I:** shows th ip address of the raspberrypi.
g. **lsusb;** lists the usb hardware connected to raspberrypi.
h. **vcgencmd measure_temp:** shows the temperature of the cpu.
i. **vcgencmd get_mem arm && vcgencmd get_mem gpu:** shows the memory split between the cpu and gpu.

5.11) Programming RaspberryPI with python

Program	Code
Print hello world	print("hello world")
program to add two numbers code	a=1.2 b=5.3 sum=float(a)+float(b) print("the sum of {0} and {1} is {2}".format(a,b,sum))
program to roll a dice	import random min=1 max=6 roll_again="yes"
	while roll_again=="yes" or roll_again=="y" print("rolling the dices") print("the values are") print(random.randomint(min,max) print(random.randint(min,max)
program to find the ip address of raspberrypi	import urllib import re print("we will try to open this url, in order to get ip address") url=http://checkip.dyndns.org print(url)
program to generate password	import string from random import* characters=string.ascii_letters+string.punctuation+string.digits password="".join(choice(charcters) for x in range(randint(8,16))) print(password)
program to print fibonacci series	a,b=0,1 while b<200: print(b) a,b=b,a+b

program to check for armstrong number	num=int(input("enter a number:"))
	initial_sum=0
	temp=num
	while temp>0:
	digit=temp%10
	initial_sum+=digit**3
	temp//=10
	if num==initial_sum:
	print(num,"is an armstrong number")
	else:
	print(num,"is not an armstrong number")

program to display calendar of given month of the year	import calendar yy=2017 mm=11 print(calendar.month(yy,mm))

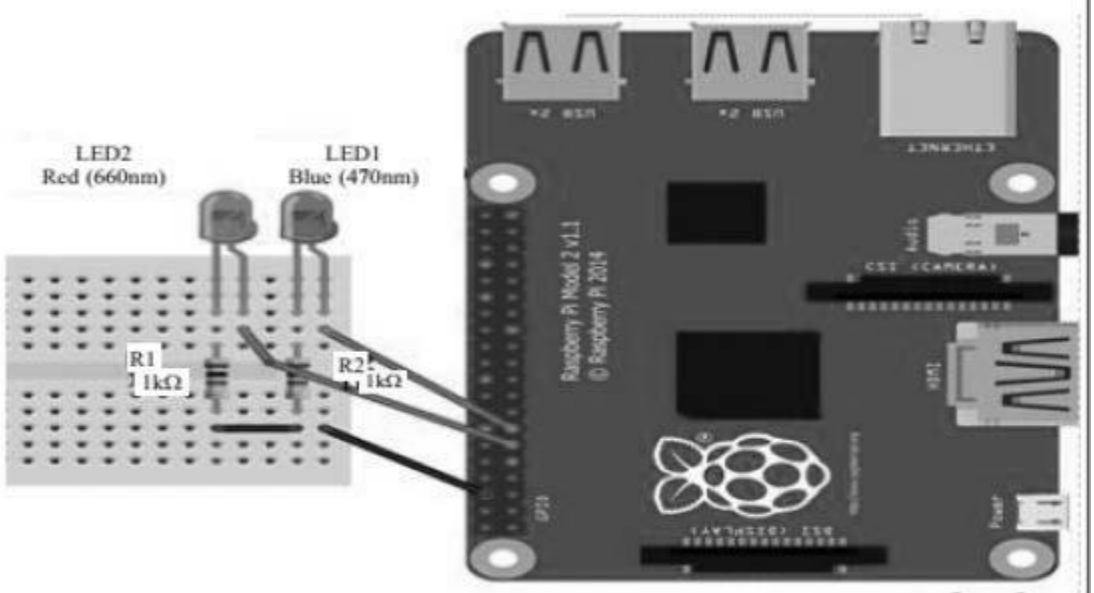
Program #1	Printing to a terminal
Components required	Raspberry Pi + SD card, Monitor + HDMI Cable, Keyboard & Mouse and Power supply
Description	Now let us start with a basic example of printing a message "Hello World" using Python Programming. Box shows how to print a greeting message to the console.
Key points	1. Find your customized Raspberry Pi. 2. Mount the SD card. 3. Plug in the HDMI cable into the Pi and the monitor. 4. Plug in the keyboard into the USB ports 5. Plug in the mouse into the USB ports 6. Plug in the power cable 7. Type in user name "pi" 8. Type in password "raspberry" 9. Double click on "Terminal" 10. This will load the "terminal" 11. Type the follow commands ✓ Change the directory by the command $ cd Desktop ✓ Create a new directory $ mkdir python_code ✓ Change the directory to python_code $ cd python_code ✓ create new file helloworld.py ✓ Now enter the code given in the box below ✓ Run the python code "sudo python helloworld.py ✓ You will see it print "Hello World!" to the screen

Code	**File:Helloworld.py**
	#Access the python working environment
	#!/usr/bin/python
	#Print a message Hello world on to the terminal
	print("Hello World!")

Output	A message "Hello world" will prints on the console.

Program #2	Blinking an LED
Components required	Raspberry Pi + SD card, Monitor + HDMI Cable, Keyboard & Mouse and Power supply, 1 Red LED and Blue LED, 2 1K resistors.
Description	Now let us look at an example of controlling the state of LEDs from ON to OFF state or vice versa from Raspberry Pi. Figure shows the schematic diagram of connecting an LEDs to Raspberry Pi. Box shows how to turn the LED on/off from by running the code given. In this example the LEDs are connected to GPIO pin 17 and 27 respectively which is initialized as output pin. The state of the LED is toggled by the executing the two programs given in the Box. Box
	having the code changes the state of the LED twice whereas Box having the code runs an infinite loop to flicker LEDs from ON/OFF state every second. Run the code given below and observe at the desired output.

Circuit Diagram



Key points	1. Create file "blink.py" 2. Create file "blink_ever.py" 3. Enter the above code 4. Run the python file "sudo python blink.py" << Watch the LEDs blink 2 times
	5. Run the python file "sudo python blink_ever.py" << Watch the LEDs blink forever
Code	**File: blink.py**

```
#Access the python working environment
#!/usr/bin/python
#import the time module so as to switch LEDs on/off with the time elapsed
#import the RPI.GPIO library
import RPi.GPIO as GPIO
#use one of the two numbering system either BOARD numbers/BCM
# Refer to the channel numbers on the Broadcom SOC.
GPIO.setmode(GPIO.BCM)
#Configure Pin 17 as an OUTPUT
GPIO.setup(17,GPIO.OUT)
#Configure Pin 27 as an OUTPUT
GPIO.setup(27,GPIO.OUT)
#Turn up LEDs on pin 17
GPIO.output(17,GPIO.HIGH)
#Turn up LEDs on pin 27
GPIO.output(27,GPIO.HIGH)`
#wait for 1 second
time.sleep(1)
#Turn up LEDs off on pin 17
GPIO.output(17,GPIO.LOW)
#Turn up LEDs off on pin 27
GPIO.output(27,GPIO.LOW)
#wait for 1 second
time.sleep(1)
```

File:blink_ever.py

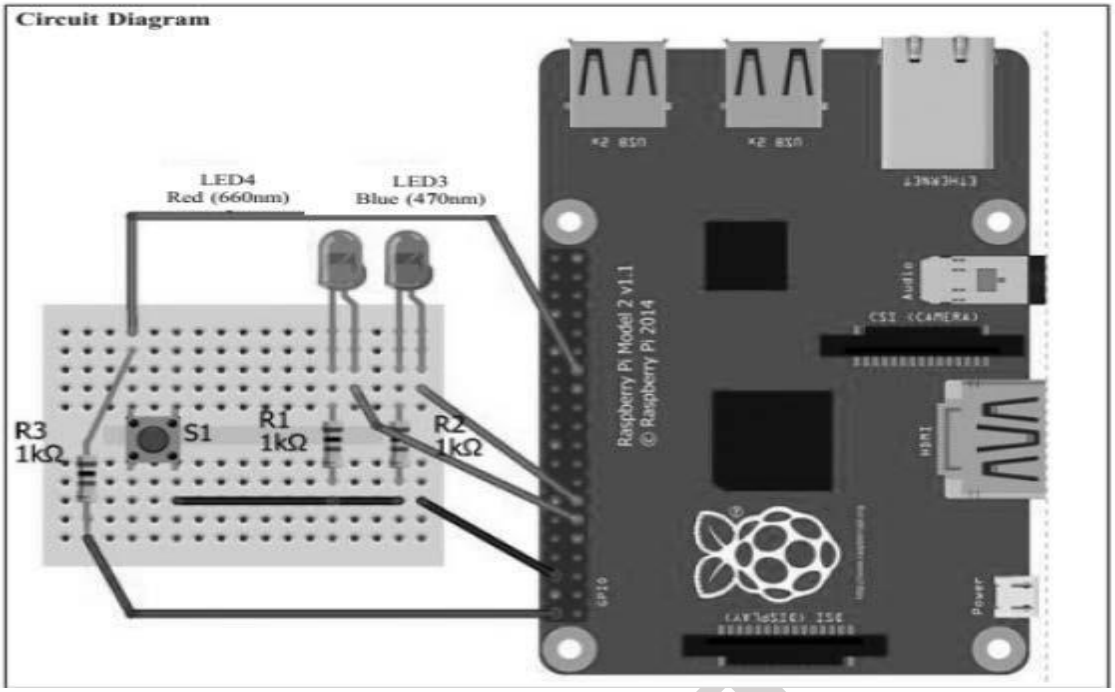
```
#Access the python working environment
#!/usr/bin/python
#import the time module so as to switch LEDs on/off with the time elapsed
import time
#import the RPI.GPIO library
```

```
import RPi.GPIO as GPIO
#use one of the two numbering system either BOARD numbers/BCM
# Refer to the channel numbers on the Broadcom SOC.
GPIO.setmode(GPIO.BCM)
#Configure pin 17 and 27 to be an OUTPUT pins
GPIO.setup(17,GPIO.OUT)
GPIO.setup(27,GPIO.OUT)
#Use while construct which runs infinite number of times there by blinking
        LEDs forever
while 1:
        #Turn up LEDs on
      GPIO.output(17,GPIO.HIGH)
       GPIO.output(27,GPIO.HIGH)
      time.sleep(1)
       #Turn up LEDs off
      GPIO.output(17,GPIO.LOW)
      GPIO.output(27,GPIO.LOW)
      time.sleep(1)
```

Output	LEDs turns on/off twice when blink.py file is executed and LEDs keeps changing their state forever when blink_forever.py file is executed.

Program #3	**Push button for physical input**
Components required	Raspberry Pi + SD card, Monitor + HDMI Cable, Keyboard & Mouse and Power supply, 1 Red LED and Blue LED, 2 1K resistors, push button and jumper wires.
Description	Now let us look at an example involving LEDs and a switch that is used to control LED. Figure shows the schematic diagram of connecting an LEDs and a switch to Raspberry Pi. Box shows a python program for controlling an LED with a switch.In the infinite while loop the value of pin 10 is checked and the state of the LED is toggled if the switch is pressed. This example shows how to get input from GPIO pins and process the state of the LEDS. Run the code given below and observe at the desired output.

Circuit Diagram



Key points	1. Create file "button.py" 2. Enter the above code 3. To run the python code "sudo python button.py"
Code	File: button.py #Access the python working environment #!/usr/bin/python #Import os module to enable interrupts from a push button import os #import the time module so as to know the time the user as given the input from a push button

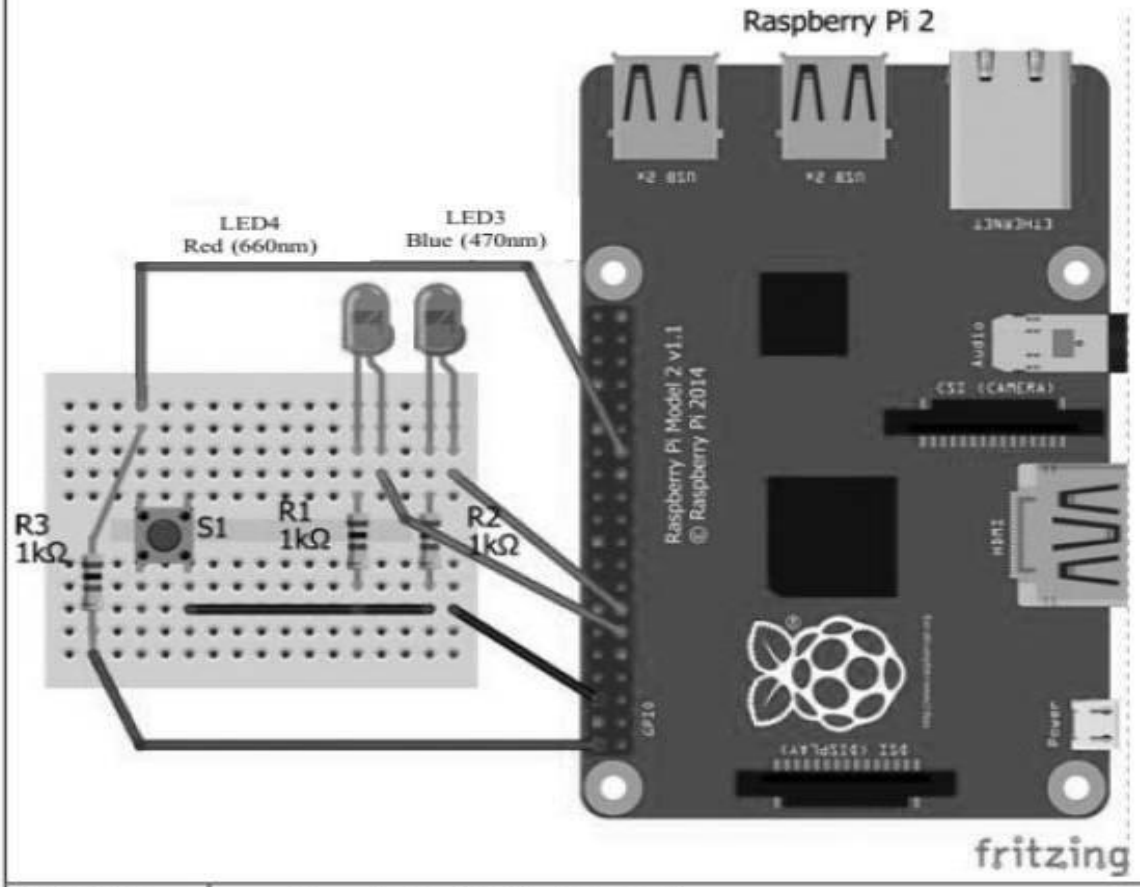
```
import time
#import the RPI.GPIO library
import RPi.GPIO as GPIO
#use one of the two numbering system either BOARD numbers/BCM
# Refer to the channel numbers on the Broadcom SOC.
GPIO.setmode(GPIO.BCM)
#configure pin 10 to be INPUT which reads the status of a switch button
GPIO.setup(10, GPIO.IN)
#print a message on to the terminal
print(" Button + GPIO ")
#Read the status of a button from GPIO pin 10
print GPIO.input(10)
#Run a infinite loop on the status of the button read
while True:
  if ( GPIO.input(10) == True ):
      print("Button Pressed")
     #Print the time when the input was given from the push button
      os.system('date')
     #Read the status of a button from GPIO pin 10
      print GPIO.input(10)
     #wait for 5 seconds
      time.sleep(5)
```

```
else:
    #clear the system variables
    os.system('clear')
    #prompt the user to give an input
    print ("Waiting for you to press a button")
time.sleep(1)
```

Output	Press the Push button switch to Turn ON/OFF the LED

Program #4	**Interact with the user**
Components required	Raspberry Pi + SD card, Monitor + HDMI Cable, Keyboard & Mouse and Power supply, 1 Red LED and Blue LED, 2 1K resistors, push button and jumper wires.
Description	Now let us look at an example involving LEDs and a switch that is used to control LED depending on what a user choose. Figure shows the schematic diagram of connecting an LEDs and a switch to Raspberry Pi. Box shows a python program for controlling an LED by reading two input values, one for which LED would user like to blink(option 1-for Red, 2- for Blue) and one more parameter to set the maximum number of times the LED should be flickered. This example shows how to get input from a user and process the state of the LEDS. Run the code given below and observe at the desired output.

Circuit Diagram



Key points	1. Create file "user_input.py" 2. Enter the above code

	3. Run the python file by "sudo python user_input.py" << Run through the questions and make an LED blink.

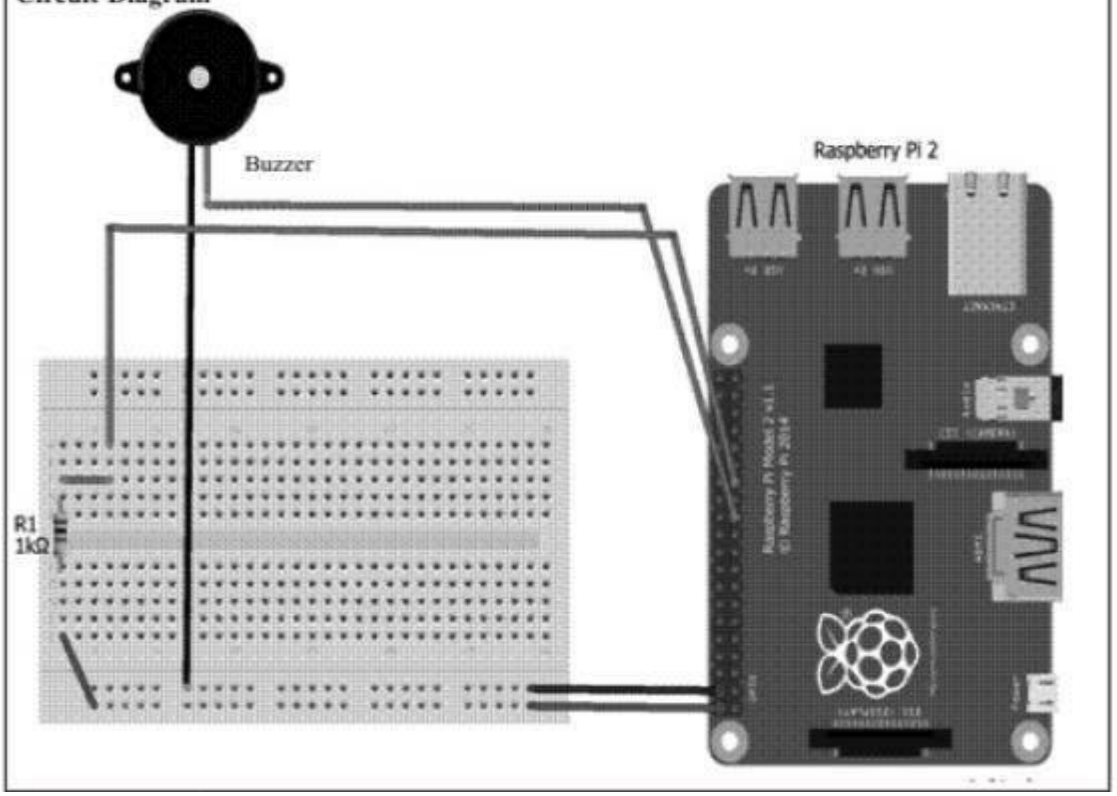
Code	**File: user_input.py**
	#Access the python working environment
	#!/usr/bin/python
	#Import os module to enable interrupts from a push button
	import os
	#import the time module so as to switch LEDs on/off with the time elapsed
	import time
	#import the RPI.GPIO library
	import RPi.GPIO as GPIO
	#use one of the two numbering system either BOARD numbers/BCM
	# Refer to the channel numbers on the Broadcom SOC
	GPIO.setmode(GPIO.BCM)
	#configure pin 17 to be OUTPUT pin
	GPIO.setup(17,GPIO.OUT)
	#configure pin 27 to be OUTPUT pin
	GPIO.setup(27,GPIO.OUT)
	
	#Initialize variables for user input
	led_ch = 0
	counter = 0
	
	#clear the python interpreter console
	os.system('clear')
	
	print "Which LED would you like to blink"
	#Accept 1 for Red
	print "1: Red?"
	#Accept 2 for Blue
	print "2: Blue?"
	
	led_ch = input("Choose your option: ")
	
	if led_ch == 1:
	#clear the python interpreter console
	os.system('clear')
	print "You picked the Red LED"
	counter = input("How many times would you like it to blink?: ")
	while counter > 0:
	#on LED on Pin 27
	GPIO.output(27,GPIO.HIGH)
	time.sleep(1)
	#off LED on pin 27
	GPIO.output(27,GPIO.LOW)
	time.sleep(1)
	#Record the number of counts on LED
	counter = counter – 1

```
if led_ch== 2:
        #clear the python interpreter console
        os.system('clear')
        print "You picked the Red LED"
        counter = input("How many times would you like it to blink?: ")
        while counter > 0:
                #on LED Pin 27
                GPIO.output(17,GPIO.HIGH)
                time.sleep(1)
                #off LED on pin 27
                GPIO.output(17,GPIO.LOW)
                time.sleep(1)
                #Record the number of counts on LED
                count = count – 1
```

Output	LED gets flickered on the inputs given by the user.

Program #5	**Buzzer**
Components required	Raspberry Pi + SD card, Monitor + HDMI Cable, Keyboard & Mouse and Power supply, 1 Red LED and Blue LED, 2 1K resistors, push button and jumper wires, Breadboard, buzzer.
Description	Now let us look at an example involving a Piezo buzzer and a switch that is used to beep a number of times the user choose. Figure shows the schematic diagram of connecting an piezo buzzer to pin 22 and a switch to Raspberry Pi. Box shows a python program for controlling an piezo buzzer by reading an input value which runs over a loop to beep number of times the user has choose. Run the code given below and observe at the desired output.

Circuit Diagram



Key points	1. Create file "buzzer.py" 2. Enter the code above code 3. To run python code "sudo python buzzer.py"<<listen to it beep

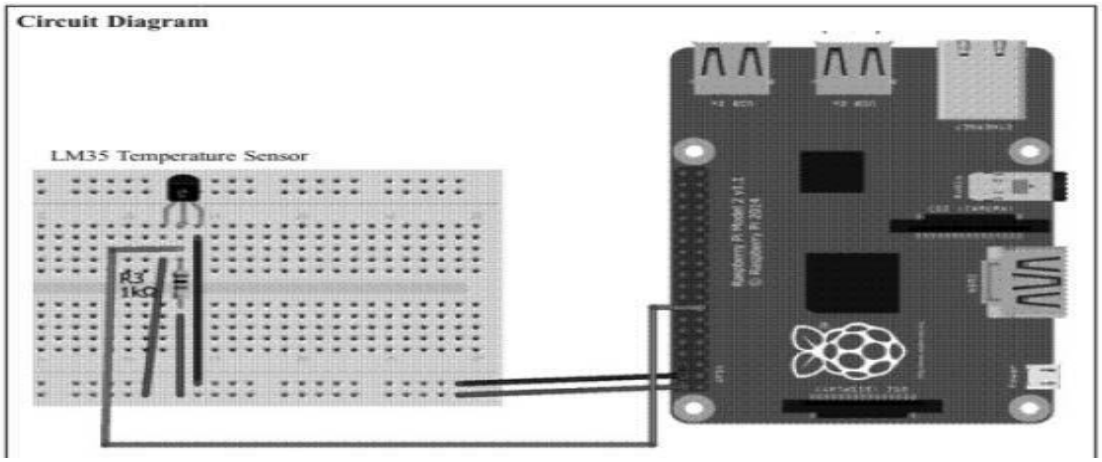
Code	**File: buzzer.py**
	#!/usr/bin/python
	import os
	import time
	import RPi.GPIO as GPIO
	GPIO.setmode(GPIO.BCM)
	GPIO.setwarnings(False)
	GPIO.setup(22,GPIO.OUT)
	loop_counter = 0
	def morsecode ():
	#Dot Dot Dot
	GPIO.output(22,GPIO.HIGH)
	time.sleep(.1)
	GPIO.output(22,GPIO.LOW)
	time.sleep(.1)
	GPIO.output(22,GPIO.HIGH)
	time.sleep(.1)
	GPIO.output(22,GPIO.LOW)
	time.sleep(.1)
	GPIO.output(22,GPIO.HIGH)
	time.sleep(.1)
	#Dash Dash Dah
	GPIO.output(22,GPIO.LOW)
	time.sleep(.2)
	GPIO.output(22,GPIO.HIGH)
	time.sleep(.2)
	GPIO.output(22,GPIO.LOW)
	time.sleep(.2)
	GPIO.output(22,GPIO.HIGH)
	time.sleep(.2)
	GPIO.output(22,GPIO.LOW)
	time.sleep(.2)
	GPIO.output(22,GPIO.HIGH)
	time.sleep(.2)
	GPIO.output(22,GPIO.LOW)
	time.sleep(.2)

```
            #Dot Dot Dot
            GPIO.output(22,GPIO.HIGH)
            time.sleep(.1)
            GPIO.output(22,GPIO.LOW)
            time.sleep(.1)
            GPIO.output(22,GPIO.HIGH)
            time.sleep(.1)
            GPIO.output(22,GPIO.LOW)
            time.sleep(.1)
            GPIO.output(22,GPIO.HIGH)
            time.sleep(.1)
            GPIO.output(22,GPIO.LOW)
            time.sleep(.7)
os.system('clear')
print "Morse Code"
loop_count = input("How many times would you like SOS to loop?: ")
while loop_count > 0:
            loop_counter = loop_counter - 1
            morsecode ()
```

Output	listen to beep of piezo buzzer

Program #6	**Temperature sensor**
Components required	Raspberry Pi + SD card, Monitor + HDMI Cable, Keyboard & Mouse and Power supply, 1 Red LED and Blue LED, 2 1K resistors, push button and jumper wires, Breadboard, buzzer.1 LM35 temperature sensor.
Description	Now let us look at an example involving an DS18B22 temperature sensor which reads out a temperature and records on to a terminal. Figure shows the schema -tic diagram of connecting an DS18B22 temperature sensor to Raspberry Pi board. Box shows a python program which records the temperature read by LM35 temperature sensor.This example shows how to get an analog input from GPIO pins and process the input. An infinite loop runs over the sensor which records a temperature every second Compile the code given below and upload it to Arduino UNO Board to observe at the desired output.

Circuit Diagram



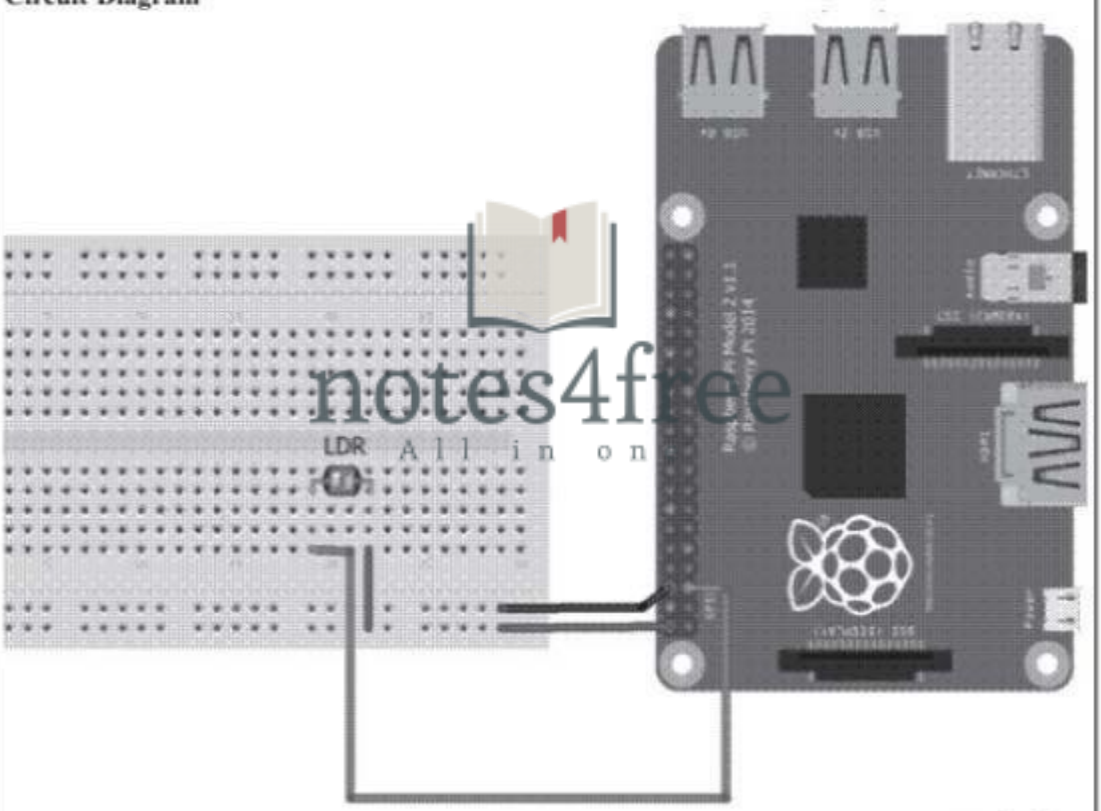
Code

```
import os
import glob
import time
#initialize the device
os.system('modprobe w1-gpio')
os.system('modprobe w1-therm')

base_dir = '/sys/bus/w1/devices/'
device_folder = glob.glob(base_dir + '28*')[0]
device_file = device_folder + '/w1_slave'

def read_temp_raw():
    f = open(device_file, 'r')
   lines = f.readlines()
    f.close()
   return lines

def read_temp():
    lines = read_temp_raw()
   while lines[0].strip()[-3:] != 'YES':
        time.sleep(0.2)
       lines = read_temp_raw()
    equals_pos = lines[1].find('t=')
    if equals_pos != -1:
       temp_string = lines[1][equals_pos+2:]
        temp_c = float(temp_string) / 1000.0
       temp_f = temp_c * 9.0 / 5.0 + 32.0
        return temp_c, temp_f

while True:
          print(read_temp())
          time.sleep(1)
```

Output Current Room Temperature is recorded.

Program #7	Light sensor
Components required	Raspberry Pi + SD card, Monitor + HDMI Cable, Keyboard & Mouse and Power supply, 1 Red LED and Blue LED, 2 1K resistors, push button and jumper wires, Breadboard, buzzer.LM35 temperature sensor, LDR Light Dependent resistor
Description	Now let us look at an example involving an LDR sensor which reads the intensity of light and records it an text file. Figure shows the schematic diagram of connecting an LDR sensor to Raspberry Pi board. Box shows a python program which records the intensity of light to a text file. This example shows how to get an analog input from GPIO pins and process the input. An infinite loop runs over the sensor which records intensity of light with date and time stamps every second Compile the code given below and upload it to Arduino UNO Board to observe at the desired output.

Circuit Diagram



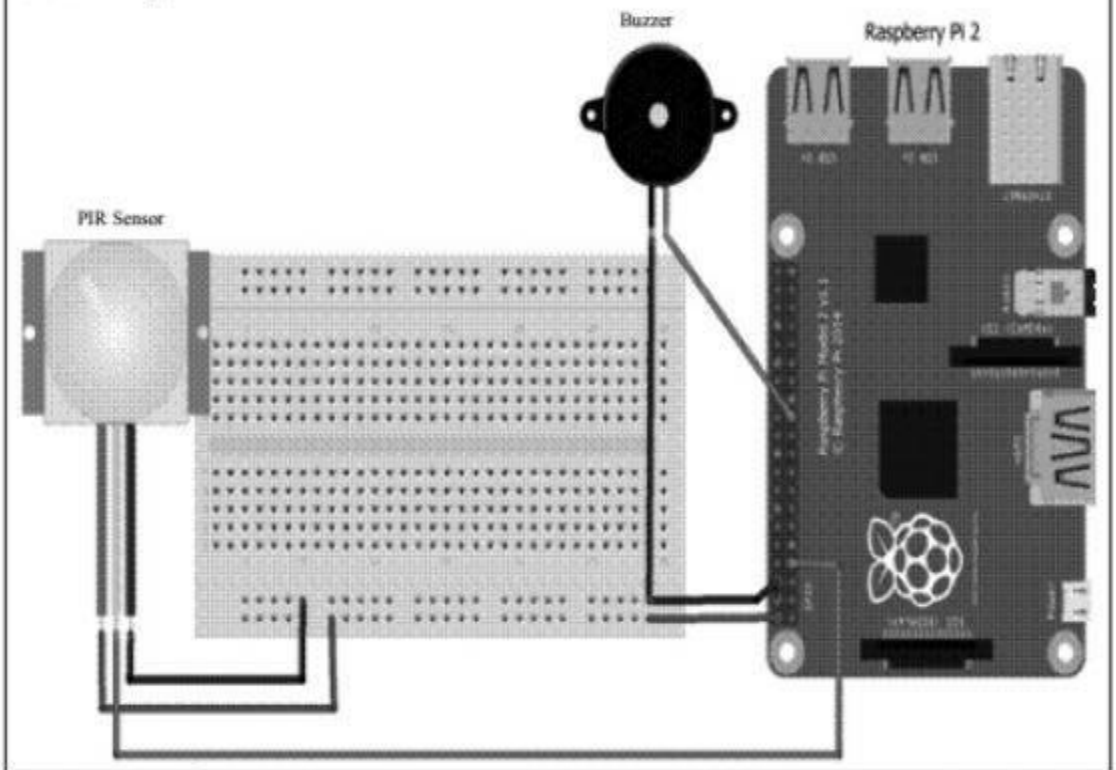
Key points	1. Create file "ldr.py" then "touch foo.txt" 2. To run the python code "sudo python ldr.py" << See what the light levels in the room are. 3. The check the file "more foo.txt" you can see your results.
Code	**File: ldr.py** #!/usr/bin/env python import os import datetime import time import RPi.GPIO as GPIO

```
DEBUGER = 1
GPIO.setmode(GPIO.BCM)
def RCtimer (RCpins):
        readings = 0
       GPIO.setup(RCpins, GPIO.OUT)
        GPIO.output(RCpins, GPIO.LOW)
       time.sleep(.1)

       GPIO.setup(RCpins, GPIO.IN)
        # iterates 1 miliseconds over one cycle
       while (GPIO.input(RCpins) == GPIO.LOW):
              readings += 1
       return readings

while True:
                    GetDateTime = datetime.datetime.now().strftime("%Y-%m-%d
%H:%M:%S")\
                    LDRReading = RCtimes(3)
                    print RCtimes(3)

                    # Open a file
                    fo = open("/home/pi/10x10/foo.txt", "wb")
                    fo.write (GetDateTime)
                    LDRReading = str(LDRReading)
                    fo.write ("\n")
                    fo.write (LDRReading)

                    # Close opend file
                    fo.close()
                    time.sleep(1)
```

Output	Intensity of the light in the room is recorded on to a terminal as well as to text file.

Program #8	**Passive Inferred Sensor**
Components required	Raspberry Pi + SD card, Monitor + HDMI Cable, Keyboard & Mouse and Power supply, 1 Red LED and Blue LED, 2 1K resistors, push button and jumper wires, Breadboard, buzzer.LM35 temperature sensor, LDR Light Dependent resistor, PIR(Passive Infrared sensor) sensor.
Description	Now let us look at an example involving an PIR sensor which detects the motion of an object. Figure shows the schematic diagram of connecting an PIR sensor to Raspberry Pi board. Box shows a python program which the motion of an object and triggers a message as "Motion detected".This example shows how to get an analog input from GPIO pins and process the input. An infinite loop runs over the PIR sensor which waits for any of the movements across its boundary. Compile the code given below and upload it to Arduino UNO Board to observe at the desired output.

Circuit Diagram



Key points	1. Create file "touch python pir.py" 2. To run the python code "sudo python pir.py" << Move in front of the PIR to activate it.
Code	**File: PIR.py**

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(27,GPIO.OUT)
GPIO_PIR_sensor = 7

print "PIR Module Test (CTRL-C to exit)"
# configure pin to be input
GPIO.setup(GPIO_PIR_sensor,GPIO.IN)

CurrentState  = 0
PreviousState = 0
```

```
try:
 print "Waiting for PIR to settle ..."
 # iterate till PIR  outputs the value 0
 while GPIO.input(GPIO_PIR_sensor)==1:
   CurrentState  = 0
 print "  Ready"
 # Iterate until user types CTRL-C
 while True :
   # status of the PIR to be read
  CurrentState = GPIO.input(GPIO_PIR_sensor)
   if CurrentState==1 and PreviousState==0:
    # Trigger action on PIR
     print "  Motion detected!"
    # Previous status of the PIR to be recorded
     GPIO.output(27,GPIO.HIGH)
    time.sleep(1)
     GPIO.output(27,GPIO.LOW)
    PreviousState=1
   elif CurrentState==0 and PreviousState==1:
    # check if PIR has arrived to the ready state
     print "  Ready"
     PreviousState=0
   # stop for 10 milliseconds
   time.sleep(0.01)
except KeyboardInterrupt:
 print "  Quit"
  # GPIO settings to be reset
   GPIO.cleanup()
```

Output	Move in front of the PIR to activate it and sensor generates a message..

5.12) Digital Temperature Sensors Vs. Analog Temperature Sensors

- Analog temp. sensors contain thermistors and temperature value is obtained from resistance value (due to change in volt)
- Digital temperature sensors are typically silicon based integrated circuits
- Unlike analog temperature sensors, calculations are performed by the sensor, and the output is an actual temperature value

5.13) Raspberry Pi and DS18B20 Temperature Sensor

About the DS18B20

- The DS18B20 also has an alarm function that can be configured to output a signal when the temperature crosses a high or low threshold that's set by the user

- A 64 bit ROM stores the device's unique serial code. This 64 bit address allows a microcontroller to receive temperature data from many sensors with identity.
- The DS18B20 temperature sensor is perfect for projects like weather stations and home automation systems
- The size is same as a transistor and use only one wire for the data signal
- Extremely accurate and take measurements quickly

DS18B20 : Technical Specifications

- -55°C to 125°C -: temperature range
- 3.0V to 5.0V :- operating voltage
- 0.5°C (9 bit); 0.25°C (10 bit); 0.125°C (11 bit); 0.0625°C (12 bit) resolution
- 64 bit unique address
- One-Wire communication protocol

DS12B20 Pin Diagram

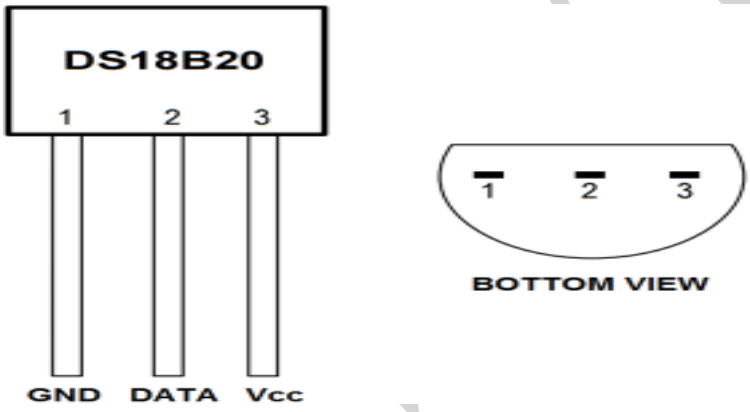


Figure 5.10 DS12B20 Pin Diagram

5.13.1) Configuring Raspberry Pi

- sudo nano /boot/config.txt
- Add this "dtoverlay=w1-gpio" at the end of file if doesn't exist
- sudo reboot
- sudo modprobe w1-gpio

Z

- sudo modprobe w1-therm
- cd /sys/bus/w1/devices
- ls
- cd 28-xxxx
- cat w1_slave
- a3 01 4b 46 7f ff 0e 10 d8 : crc=d8 YES
- a3 01 4b 46 7f ff 0e 10 d8 t=32768

5.14) Pi Via SSH (Secure Shell)

- You can access the command line of a Raspberry Pi remotely from another computer or device
- The Raspberry Pi will act as a remote device: you can connect to it using a client on another machine.

1. Set up your local network and wireless connectivity

- Make sure your Raspberry Pi is properly set up and connected.
- If you are using wireless networking, this can be enabled via the desktop's user interface, or using the command line
- You will need to note down the IP address of your Pi in order to connect to it later.
- Using the ifconfig command will display information about the current network status, including the IP address, or you can use hostname -I to display the IP addresses associated with the device

2. Enable SSH

- Enter sudo raspi-config in a terminal window
- Select Interfacing Options
- Navigate to and select SSH
- Choose Yes
- Select Ok
- Choose Finish

Z

Alternatively, use systemctl to start the service

- sudo systemctl enable ssh
- sudo systemctl start ssh

3. Enable SSH on a headless Raspberry Pi (add file to SD card on another machine)

- For headless setup, SSH can be enabled by placing a file named ssh, without any extension, onto the boot partition of the SD card from another computer.
- When the Pi boots, it looks for the ssh file. If it is found, SSH is enabled and the file is deleted

4. Set up your client

- SSH is built into Linux distributions and Mac OS. For Windows and mobile devices, third-party SSH clients are available.

5.15) Smart and Connected Cities

An IoT Strategy for Smarter Cities

- Managing a city bears some resemblance to managing a corporate enterprise.
- As the need for efficiency increases, new tools help increase operational efficiency.
- For cities, just as for businesses, digitization transforms the perspective on operations.
- New ideas emerge, bringing different approaches to solving management issues.

Vertical IoT Needs for Smarter Cities

- There are many differing approaches and solutions for city management.
- Allthese solutions typically start at the street level, with sensors that capture data on everything from parking space availability to water purity.
- Data analytics is also used extensively—for example, to reduce crime or improve traffic flows.
- Citizens can use tools to leverage their smart mobile devices, such as to report problems and make recommendations for improving urban life or locate available parking spaces.
- When enabled through connectivity, these smart solutions can have a transformative impact on quality of life.
- A recent Cisco study, as illustrated in Figure 5.11, expects IoT to have the following economic impact over a 10-year period:

Z

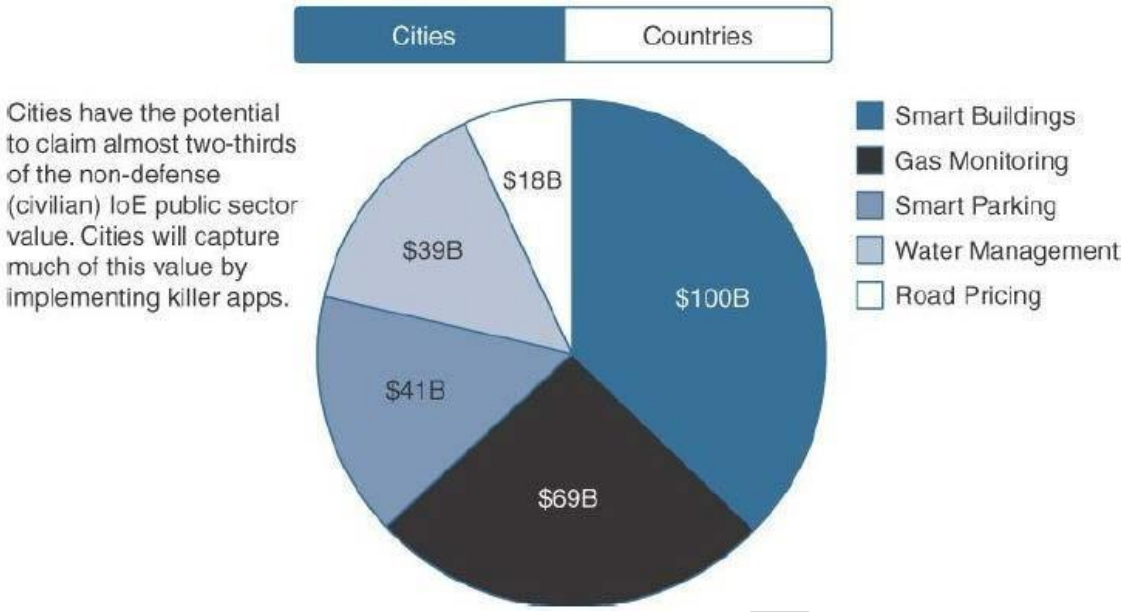


Figure 5.11 Key Use Cases for Smart Cities

1) Smart buildings: Smart buildings have the potential to save $100 billion by lowering operating costs by reducing energy consumption through the efficient integration of heating, ventilation, and air-conditioning (HVAC) and other building infrastructure systems.

- Note that the financial gain applies to city budgets only when a building is city owned.
- However, the reduced emissions benefit the city regardless of who owns the buildings.

2) Gas monitoring: Monitoring gas could save $69 billion by reducing meter-reading costs and increasing the accuracy of readings for citizens and municipal utility agencies.

- The financial benefit is obvious for users and utility companies when the utility is managed by the city.
- There are also very important advantages in terms of safety, regardless of who operates the utility.
- In cases of sudden consumption increase, a timely alert could lead to emergency response teams being dispatched sooner, thus increasing the safety of the urban environment.

3) Smart parking: Smart parking could create $41 billion by providing realtime visibility into parking space availability across a city.

- Residents can identify and reserve the closest available space, traffic wardens can identify noncompliant usage, and municipalities can introduce demand based pricing.

4) Water management: Smart water management could save $39 billion connecting household water meters over an IP network to provide remote usage and status information.

- The benefit is obvious, with features such as real-time consumption visibility and leak detection.
- A gate or a pump can be opened and closed remotely and automatically in real time, based on a variety of flow input and output analytics data.

Z

- Vibrations can be measured to detect and predict potential equipment failures.
- Repair teams can be dispatched proactively before equipment failure occurs.

Road pricing: Cities could create $18 billion in new revenues by implementing automatic payments as vehicles enter busy city zones while improving overall traffic conditions.

- Real-time traffic condition data is very valuable and actionable information that can also be used to proactively reroute public transportation services or private users.

Global vs. Siloed Strategies

- The main obstacle in implementing smart solutions in today's traditional infrastructure is the complexity of how cities are operated, financed, regulated and planned.
- Cities attempting to upgrade their infrastructure to match the growing needs of the citizen population often invest in one problem at a time, and they do it independently.
- Even cities using IoT technology break up city assets and service management into silos that are typically unable to communicate or rely on each other.

The independent investment model results in the following problems:

- Isolation of infrastructure and IT resources
- No sharing of intelligence and information, such as video feeds and data from sensors.
- Waste and duplication in investment and effort
- Difficulty scaling infrastructure management

- All these requirements pose technological challenges, including the following:
  - How do you collect the data? What are the various sources of data, including hardware endpoints and software?
  - How do you make sure that any data collection devices, such as sensors, can be maintained without high costs?
  - Where do you analyze the data? What data do you carry back to the cloud, and what data do you analyze locally?
  - What kind of network connectivity is best suited for each type of data to collect?
  - What kind of power availability and other infrastructure, such as storage, is required?
  - How do you aggregate data from different sources to create a unified view?
  - How do you publish the data and make it available for applications to consume?
  - How do you make the end analysis available to specialized smart city personnel, such as traffic operators, parking enforcement officers, street lighting operators, and so on at their logical decision points?
  - How do you present the long-term analysis to city planners?

Z

5.16) Smart City IoT Architecture

- A smart city IoT infrastructure is a four-layered architecture, as shown in Figure 5-12.
- Data flows from devices at the street layer to the city network layer and connect to the data center layer, where the data is aggregated, normalized, and virtualized.
- The data center layer provides information to the services layer, which consists of the applications that provide services to the city.

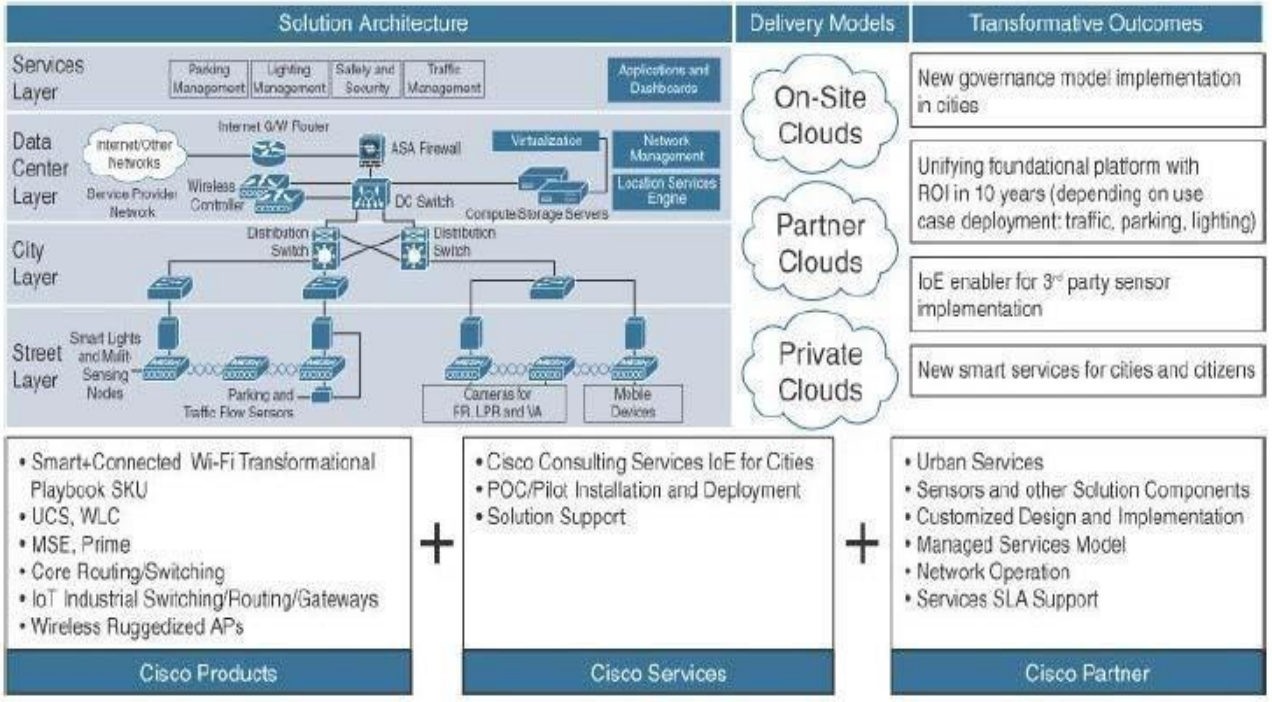


Figure 5.12 Smart Cities Layered Architecture

1) Street Layer

- The street layer is composed of devices and sensors that collect data and take action based on instructions from the overall solution, as well as the networking components needed to aggregate and collect data.
- A sensor is a data source that generates data required to understand the physical world. Sensor devices are able to detect and measure events in the physical world.
- ICT (information and communication technology) connectivity solutions rely on sensors to collect the data from the world around them so that it can be analyzed and used to operationalize use cases for cities.
- A variety of sensors are used at the street layer for a variety of smart city use cases
    - A magnetic sensor can detect a parking event by analyzing changes in the surrounding magnetic field when a heavy metal object, such as a car or a truck, comes close to it (or on top of it).
    - A lighting controller can dim and brighten a light based on a combination of time-based and ambient conditions.

Z

- Video cameras combined with video analytics can detect vehicles, faces, and traffic conditions for various traffic and security use cases.
- An air quality sensor can detect and measure gas and particulate matter concentrations to give a hyper-localized perspective on pollution in a given area.
- Device counters give an estimate of the number of devices in the area, which provides a rough idea of the number of vehicles moving or parked in a street or a public parking area, of pedestrians on a sidewalk, or even of birds in public parks or on public monuments—for cities where bird control has become an issue.

2) City Layer

- At the city layer, which is above the street layer, network routers and switches must be deployed to match the size of city data that needs to be transported.
- This layer aggregates all data collected by sensors and the end-node network into a single transport network.
- The city layer may appear to be a simple transport layer between the edge devices and the data center or the Internet.
- However, one key consideration of the city layer is that it needs to transport multiple types of protocols, for multiple types of IoT applications.
- Some applications are delay- and jitter-sensitive, and some other applications require a deterministic approach to frame delivery.
- As a result, the city layer must be built around resiliency, to ensure that a packet coming from a sensor or a gateway will always be forwarded successfully to the headend station. Figure 5.13 shows a common way of achieving this goal.

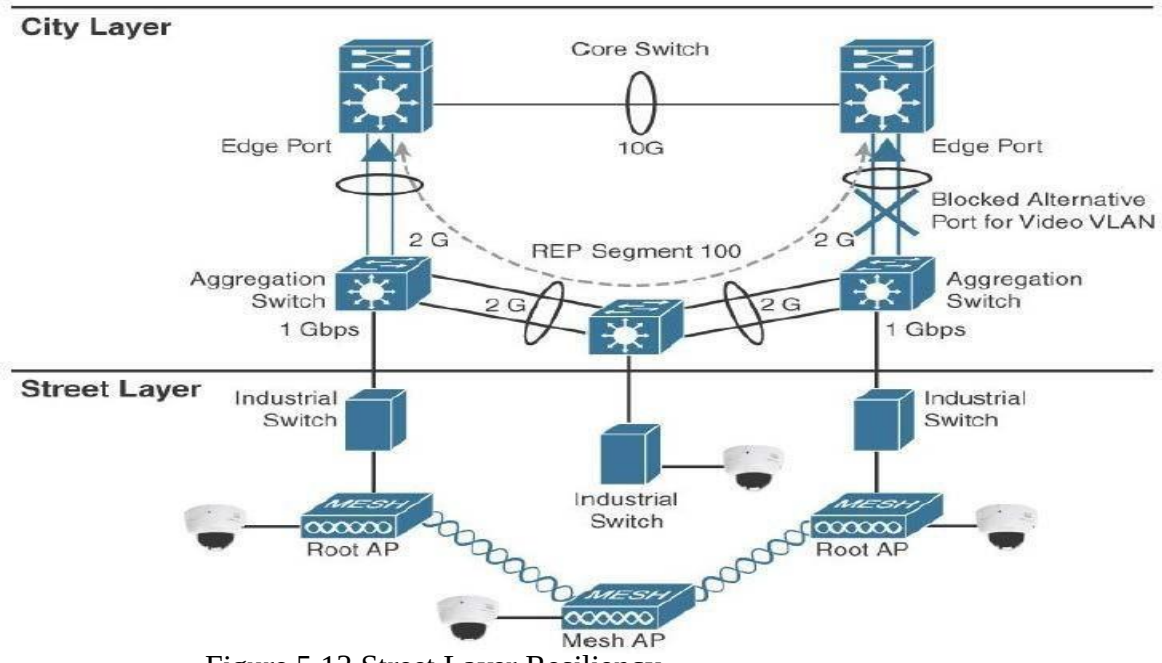


Figure 5.13 Street Layer Resiliency

3) Data Center Layer

- Ultimately, data collected from the sensors is sent to a data center, where it can be processed and correlated.
- Based on this processing of data, meaningful information and trends can be derived, and information can be provided back.
- For example, an application in a data center can provide a global view of the city traffic and help authorities decide on the need for more or less common transport vehicles.
- At the same time, an automated response can be generated.
- For example, the same traffic information can be processed to automatically regulate and coordinate the street light durations at the scale of the entire city to limit traffic congestion.
- The key technology in creating any comprehensive smart solution with services is the cloud.
- With a cloud infrastructure, data is not stored in a data center owned directly or indirectly by city authorities.
- Instead, data is stored in rented logical containers accessed through the Internet.
- Because the containers can be extended or reduced based on needs, the storage size and computing power are flexible and can adapt to changing requirements or budget conditions.
- In addition, multiple contractors can store and process data at the same time, without the complexity of exclusively owned space.
- This proximity and flexibility also facilitate the exchange of information between smart systems and allow for the deployment of new applications that can leverage information from several IoT systems.
- Figure 5.14 shows the vision of utilizing the cloud in smart solutions for cities.
- The cloud provides a scalable, secure, and reliable data processing engine that can handle the immense amount of data passing through it.
- Smart city issues require not just efficient use of infrastructure, which the cloud helps enable, they also require new data processing and management models.
- For example, cloud services allow for Software as a Service (SaaS) models that create cyclical returns on investment.

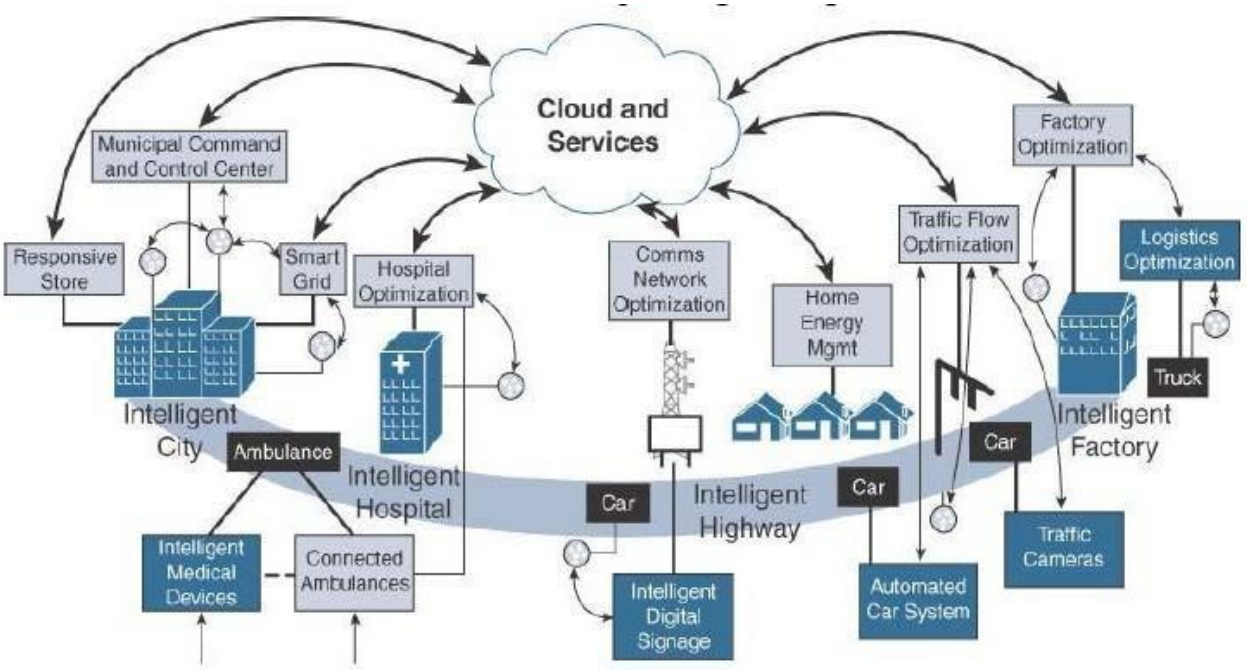


Figure 5.14 The Role of the Cloud for Smart City Applications

- With the cloud approach shown in Figure 5.14 , smart cities can also take advantage of operating expense–based consumption models to overcome any financial hurdles in adopting solutions to their most critical issues.
- Critical data, such as air condition (humidity, temperature, pollution) levels monitoring, lights. In times when city budgets are strained, data processing can be scaled down to can be processed initially.
- Then, as the efficiency of IoT is scaled up, richer data processing can be enabled in the es sential services.

4) Services Layer

- Ultimately, the true value of ICT connectivity comes from the services that the measured data can provide to different users operating within a city.
- Smart city applications can provide value to and visibility for a variety of user types, including city operators, citizens, and law enforcement.
- The collected data should be visualized according to the specific needs of each consumer of that data and the particular user experience requirements and individual use cases.
- For example, parking data indicating which spots are and aren't currently occupied can drive a citizen parking app with a map of available spots, as well as an enforcement officer's understanding of the state (utilization and payment) of the public parking space,

Z

while at the same time helping the city operator's perspective on parking problem areas in the city at any given time.

- With different levels of granularity and scale, the same data performs three different functions for three different users.
- Along the same lines, traffic information can be used by individual car drivers to find the least congested route.
- A variation of the same information can be made available to public transportation users to estimate travel times.
- Public transportation systems, such as buses, can be rerouted around known congestion points.
- The number of subway trains can be increased dynamically to respond to an increase in traffic congestion, anticipating the decisions of thousands or even millions of commuters to take public transportation instead of cars on days when roads are very congested.
- Here again, the same type of data is utilized by different types of users in different ways based on their specific use cases.

On-Premises vs. Cloud

- Different cities and regions have different data hosting requirements based on security or legal policies. A key consideration in developing ICT connectivity solutions is whether a city has requirements about where data should be hosted.
- Data can be hosted on-premises or in the cloud. Fog architectures provide an intermediate layer.
- The data resulting from fog processing can be sent to the cloud or to a data center operated locally (on-premises).
- On-premises encompasses traditional networks, and all their limitations, whereas cloud hosting encompasses a whole host of security risks if the proper measures are not taken to secure citizen data.
- When data is sent to the cloud, data sovereignty (supremacy) laws may restrict the physical location where this data is actually stored.
- Ideally, a smart city utilizing ICT connectivity would use the cloud in its architecture, but if this is impossible, the city would need to invest far more in the city layer's networking components (for example, switches, routers) and still may not be able to drive the same cross-domain value propositions and scalability in its design.

5.17) Smart City Security Architecture

- A serious concern of most smart cities and their citizens is data security.
- Vast quantities of sensitive information are being shared at all times in a layered, realtime architecture, and cities have a duty to protect their citizens' data from unauthorized access, collection, and tampering.
- In general, citizens feel better about data security when the city itself, and not a private entity, owns public or city-relevant data.
- It is up to the city and the officials who run it to determine how to utilize this data.
- When a private entity owns city-relevant data, the scope of the ownership may initially be very clear.
- However, later considerations or changes in the private entity strategy may shift the way the data is used.
- It may then be more difficult for city authorities or the citizens to oppose this new direction, simply because they do not have any stake in the decision-making process of the private entity.
- For example, suppose that a private contractor is in charge of collecting and managing parking sensor data.
- One possible way to increase the profitability of such data is to sell it to insurance companies looking to charge an additional premium to car owners parking in the street (vs. in a covered and secured garage).
- Such deviations from the original mandate are less likely to happen when cities own the data and when citizens have a way to vote against such usages.
- A security architecture for smart cities must utilize security protocols to fortify each layer of the architecture and protect city data.
- Figure 5.15 shows a reference architecture, with specific security elements highlighted. Security protocols should authenticate the various components and protect data transport throughout.
- For example, hijacking traffic sensors to send false traffic data to the system regulating the street lights may result in dramatic congestion issues.
- The benefit for the offender may be the ability to get "all greens" while traveling, but the overall result would typically be dangerous and detrimental to the city.
- The security architecture should be able to evolve with the latest technology and incorporate regional guidelines (for example, city by-laws, county or regional security regulations).

Z

- Network partners may also have their own compliance standards, security policies, and governance requirements that need to be added to the local city requirements.

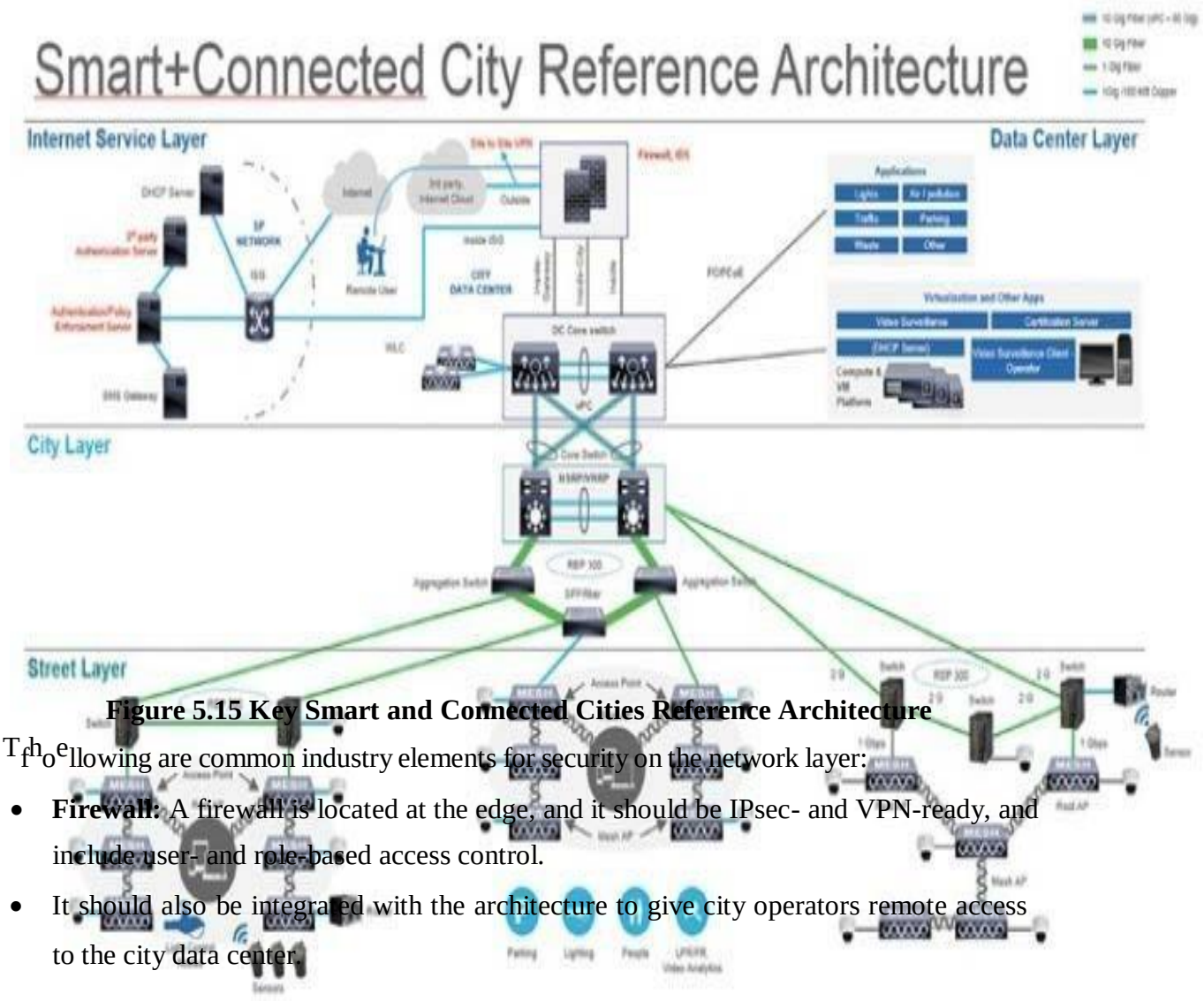


Figure 5.15 Key Smart and Connected Cities Reference Architecture

The following are common industry elements for security on the network layer:

- **Firewall:** A firewall is located at the edge, and it should be IPsec- and VPN-ready, and include user- and role-based access control.
- It should also be integrated with the architecture to give city operators remote access to the city data center.
- **VLAN:** A VLAN provides end-to-end segmentation of data transmission, further protecting data from rogue intervention. Each service/domain has a dedicated VLAN for data transmission.
- **Encryption:** Protecting the traffic from the sensor to the application is a common requirement to avoid data tampering and eavesdropping.
- In most cases, encryption starts at the sensor level. In some cases, the sensor-to gateway link uses one type of encryption, and the gateway-to-application connection uses another encryption (for example, a VPN).

Z

5.17) Smart City Use-Case Examples

- There are multiple ways a smart city can improve its efficiency and the lives of its citizens.
- The following sections examine some of the applications commonly used as starting points to implement IoT in smart cities: connected street lighting, smart parking, smart traffic control, and connected environment.

Connected Street Lighting

- Of all urban utilities, street lighting comprises one of the largest expenses in a municipality's utility bill, accounting for up to 40% of the total, according to the New York State Department of Environmental Conservation.
- Maintenance of street lights is an operational challenge, given the large number of lights and their vast geographic distribution.

Connected Street Lighting Solution

- Cities commonly look for solutions to help reduce lighting expenses and at the same time improve operating efficiencies while minimizing upfront investment.
- The installation of a smart street lighting solution can provide significant energy savings and can also be leveraged to provide additional services.
- In this regard, light-emitting diode (LED) technology leads the transition from traditional street lighting to smart street lighting:
- LEDs require less energy to produce more light than legacy lights, and they have a much longer life span and a longer maintenance cycle.
- A leading lighting company estimates that a complete switch to LED technology can reduce individual light bills by up to 70%.
- LEDs are well suited to smart solution use cases.
- For example, LED color or light intensity can be adapted to site requirements (for example, warmer color and lower intensity in city centers, sun-like clarity on highways, time- and weather-adaptive intensity and color).
- Figure 5.16 shows how electricity prices rise, while LED prices decrease and their unit sales rise.

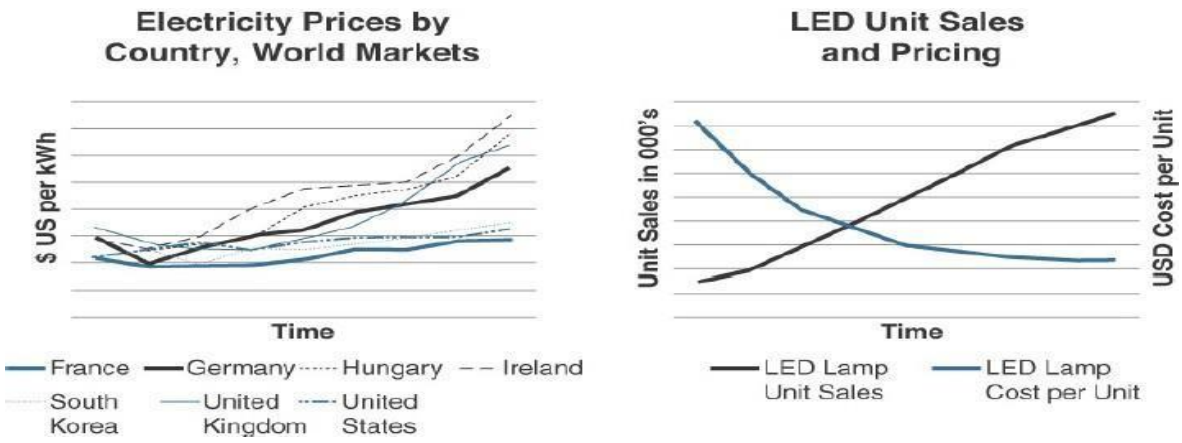


Figure 5.16 Electricity Cost vs. LED Cost and Sales

Street Lighting Architecture

- Connected lighting uses a light management application to manage street lights remotely by connecting to the smart city's infrastructure.
- This application attaches to LED lights, monitors their management and maintenance, and allows you to view the operational status of each light.
- In most cases, a sensor gateway acts as an intermediate system between the application and the lights (light control nodes).
- The gateway relays instructions from the application to the lights and stores the local lights' events for the application's consumption.
- The controller and LED lights use the cloud to connect to the smart city's infrastructure, as shown in Figure 5.17.

Z

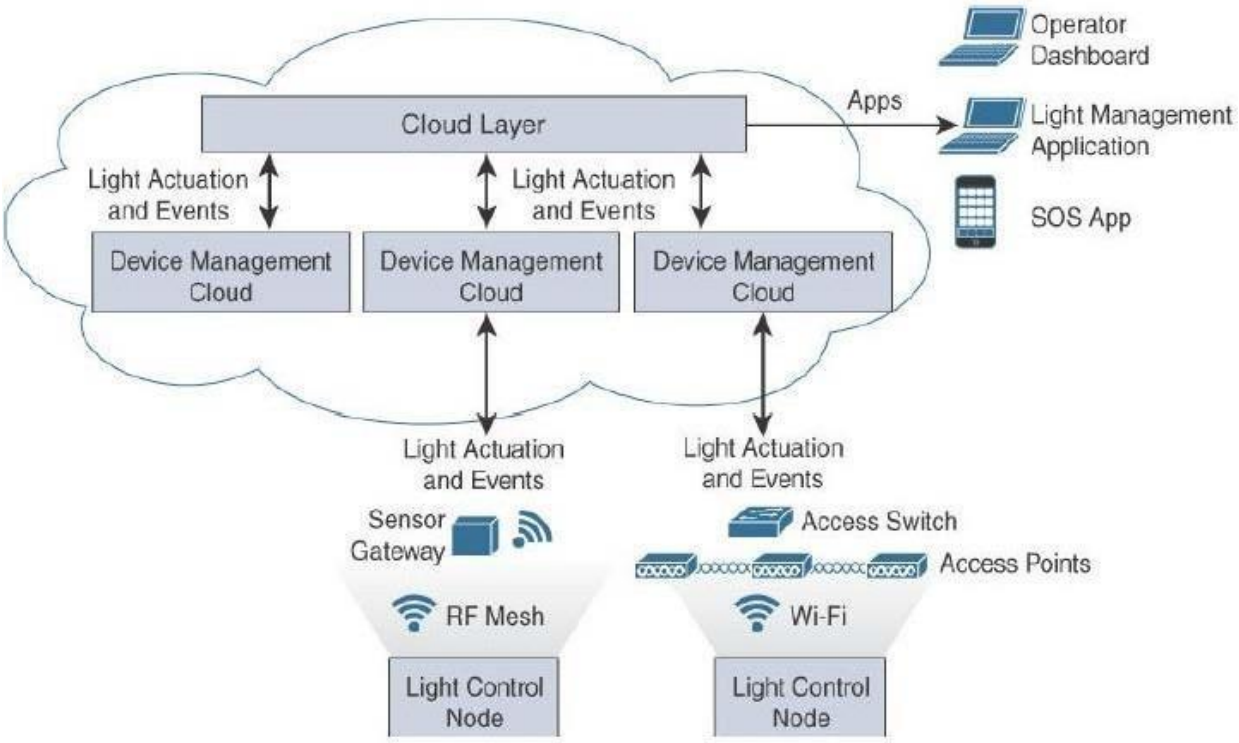


Figure 5.17 Connected Lighting Architecture

- A human or automated operator can use a cloud application to perform automated scheduling for lights and even get light sensors to perform automated dimming or brightening, as needed.
- The schedule can also impact the light intensity level and possibly the color, depending on environmental conditions, weather, time of year, time of day, location within the city, and so on.
- Lighting nodes vary widely in the industry, especially with respect to elements such as what communication protocol they use (for example, Wi-Fi, cellular, ZigBee, 802.15.4g [Wi-SUN], LoRaWAN).
- These features are optimized for different circumstances and conditions; no single lighting node can support all environments ideally.
- Many solutions leverage wired connectivity, either by using the existing city cable infrastructure or by adding a cable adjacent to the power cable.
- In cases where cabling is not practical, wireless technologies may bring interesting capabilities.
- For example, 802.15.4g controllers can be used to form a mesh and extend the network. This extension is used not only to connect other light poles but also to connect smart meters from neighboring houses.
- In all cases, the built-in versatility offered by the four-layer architecture shown in Figure

5.17 ensures that all the different types of technologies optimized to fit any city topology can be flexibly incorporated into the solution.

Smart Parking Use Cases

- Added traffic congestion is one consequence of drivers looking for parking space, and it has several consequences:
- **Contributes to pollution:** Tons of extra carbon emissions are released into the city's environment due to cars driving around searching for parking spots when they could be parked.
- **Causes motorist frustration:** In most cities, parking spot scarcity causes drivers to lose patience and waste time, leading to road rage, inattention, and other stress factors.
- **Increases traffic incidents:** Drivers searching for parking spots cause increased congestion in the streets and that, in turn, causes increased accidents and other traffic incidents.
- Revenue loss is another consequence of drivers looking unsuccessfully for parking space, and it also has various negative side effects:
- **Cities often lose revenue:** As a result of inadequate parking meter enforcement and no-parking, no-standing, and loading-zone violations, cities lose revenue.
- **Parking administration employee productivity suffers:** Employees waste time roaming the streets, attempting to detect parking rules offenders.
- **Parking availability affects income:** Local shops and businesses lose customers because of the decreased accessibility caused by parking space shortages.

Smart Parking Architecture

- A variety of parking sensors are available on the market, and they take different approaches to sensing occupancy for parking spots.
- Examples include in-ground magnetic sensors, which use embedded sensors to create a magnetic detection field in a parking spot; video-based sensors, which detect events based on video computing (vehicle movements or presence); and radar sensors that sense the presence of vehicles (volumetric detection).
- Most sensors installed in the ground must rely on battery power, since running a power line is typically too expensive.
- In larger (for example, outdoor) environments, a longer-range Low Power Wide Area (LPWA) protocol is common, as shown in Figure 5.18.

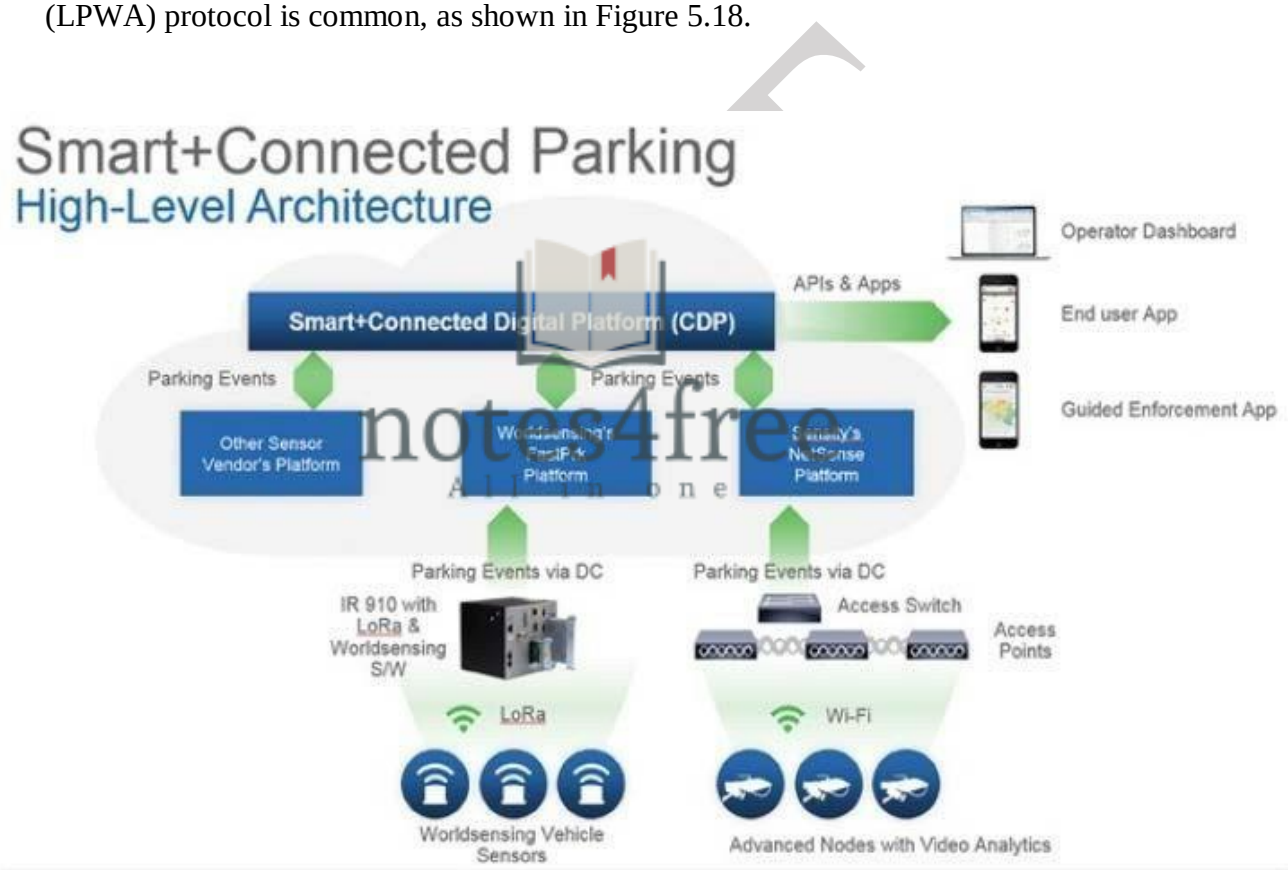


Figure 5.18 Connected Parking Architecture

- Parking sensors are typically event-driven objects.
- A sensor detects an event and identifies it based on time or analysis.
- The event is transmitted through the device's communication protocol to an access point or gateway, which forwards the event data through the city layer.
- The gateway sends it to the cloud or a fog application, where it is normalized.
- An application shows the parking event on operator dashboards, or personal smart phones, where an action can be taken.

Z

- For example, a driver can book a nearby parking spot, or a parking operator can remove it from the list of available parking spaces in target locations.
- This action triggers data to be sent back to the parking sensor to modify its availability status based on the received instructions.
- smart parking has three users that applications must support through aggregated data: city operators, parking enforcement personnel, and citizens.
- The true value of data normalization is that all parking data, regardless of technology or vendor, would be visible in these applications for the different users to support their particular experiences.
- The following are some potential user experiences for these three user types.

1) City operators: These users might want a high-level map of parking in the city to maintain perspective on the city's ongoing parking situation.

- They would also need information on historical parking data patterns to understand congestion and pain points in order to be able to effectively influence urban planning.

2) Parking enforcement officers: These users might require real-time updates on parking changes in a certain area to be able to take immediate action on enforcement activities, such as issuing tickets or sending warnings to citizens whose time is nearing expiration.

- Their focus is driving revenue creation for the city and minimizing wasted time by performing parking monitoring and enforcement at scale

3) Citizens: These users might want an application with a map (such as a built-in parking app in their car) showing available parking spots, reservation capabilities, and online payment.

- Their focus would be on minimizing the time to get a parking spot and avoiding parking tickets.
- The application could warn when parking duration limits approach, allowing the driver to move the vehicle before the timer expires or pay a parking timer extension fee without having to go back to the vehicle.

Smart Traffic Control

- Traffic is one the most well-understood pain points for any city.
- It is the leading cause of accidental death globally, causes immense frustration, and heavily contributes to pollution around the globe.
- A smart city traffic solution would combine crowd counts, transit information, vehicle counts, and so on and send events regarding incidents on the road so that other controllers on the street could take action.

Smart Traffic Control Architecture

- In the architecture shown in Figure 5.19, a video analytics sensor computes traffic events based on a video feed and only pushes events (the car count, or metadata, not the individual images) through the network.
- These events go through the architectural layers and reach the applications that can drive traffic services.
- These services include traffic light coordination and also license plate identification for toll roads.
- Some sensors can also recognize abnormal patterns, such as vehicles moving in the wrong direction or a reserved lane. In that case, the video feed itself may be uploaded to traffic enforcement agencies.

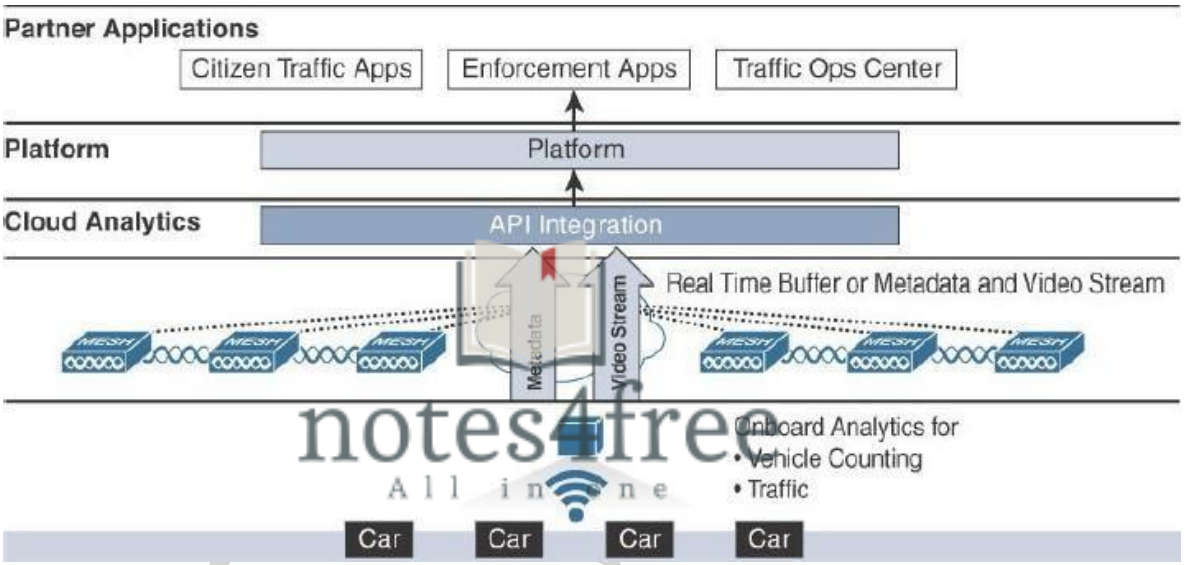


Figure 5.19 Smart City Traffic Architecture

- Other types of sensors that are part of traffic control solutions include Bluetooth vehicle counters, real-time speed and vehicle counters, and lighting control systems.
- These sensors provide a real-time perspective while also offering data collection services for historical data trending and correlation purposes.

Smart Traffic Applications

- Traffic applications can be enabled to take immediate action with other sensors to manage traffic and to reduce pain points. Historical data can be used to develop more efficient urban planning to reduce the amount of traffic a city experiences.
- A common traffic pain point is stop-and-go, where traffic flow suddenly comes to a halt and then flows again.
- This wavelike traffic pattern is a natural result of the unpredictability of the traffic speed ahead and has long been studied by public and private organizations.

- A well-known remedy for stop-and-go traffic is to regulate the standard flow speed based on car density.
- As density increases, car speed is forced down to avoid the wave effect.
- An application that measures traffic density in real time can take action by regulating the street light cycle duration to control the number of cars added to the flow of the main routes, thus limiting or suppressing the wave effect.
- From the driver's standpoint, there is a wait time before being able to get on the highway or main street, and traffic on the main route is slow but steady.
- The impression is that traffic is slow but moving, and the overall result is a better commute experience, with lowered and less stressful commute time, as well as a reduced number of accidents.

Connected Environment

- As of 2017, 50% of the world's population has settled on less than 2% of the earth's surface area.
- Such densely populated closed spaces can see spikes in dangerous gas molecules at any given moment.
- More than 90% of the world's urban population breathes in air with pollutant levels that are much higher than the recommended thresholds, and one out of every eight deaths worldwide is a result of polluted air.

The Need for a Connected Environment

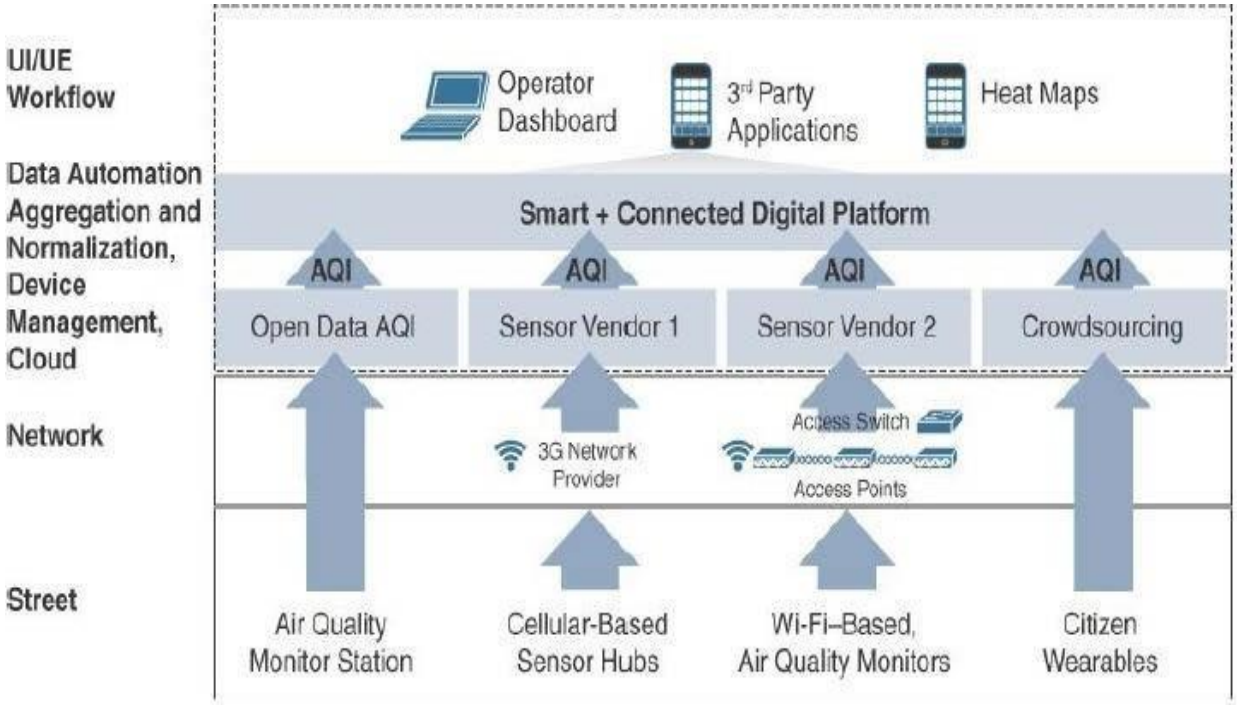
- Most large cities monitor their air quality.
- Data is often derived from enormous air quality monitoring stations that are expensive and have been around for decades.
- These stations are highly accurate in their measurements but also highly limited in their range, and a city is likely to have many blind spots in coverage.
- Given the price and size of air quality monitoring stations, cities cannot afford to purchase the number of stations required to give accurate reports on a localized level and follow the pollution flows as they move through the city over time.
- To fully address the air quality issues in the short term and the long term, a smart city would need to understand air quality on a hyper-localized, real-time, distributed basis at any given moment. To get those measurements, smart cities need to invest in the following:
    - Open-data platforms that provide current air quality measurements from existing air quality monitoring stations

Z

- Sensors that provide similar accuracy to the air quality stations but are available at much lower prices
- Actionable insights and triggers to improve air quality through cross domain actions
- Visualization of environmental data for consumers and maintenance of historical air quality data records to track emissions over time.

Connected Environment Architecture

- Figure 5.20 shows an architecture in which all connected environment elements overlay on the generalized four-layer smart city IoT architecture.



offerings, using a variety of communication protocols.

- Connected environment sensors might measure different gases, depending on a city's particular air quality issues, and may include weather and noise sensors.
- These sensors may be located in a variety of urban fixtures, such as in street lights, as explained earlier.
- They may also be embedded in the ground or in other structures or smart city infrastructure.
- Even mobile sources of information can be included through connected wearables that citizens might choose to purchase and carry with them to understand the air quality around them at any given moment.
- Crowd sourcing may make this information available to the global system.
- Communication technologies depend on the location of the sensors.

- Wearables typically communicate via a short-range technology (such as Bluetooth) with a nearby collecting device (such as a phone).
- That device, in turn, forwards the collected data to the infrastructure (for example, through cellular data).
- Sensors that are installed in urban fixtures also use a variety of communication technologies.
- Sensors included in street lighting systems may utilize the same communication infrastructure as the street light control application.
- In addition to all the air quality sensor and wearable data, the data center layer or application layer represented on the left side of Figure 5.20 also receives the open data from existing weather stations as an additional data input.
- All these data inputs come together to provide a highly accurate sense of the air quality in the city at any given moment.