

MODULE 1

What Is IoT?

IoT is a technology transition in which devices will allow us to sense and control the physical world by making objects smarter and connecting them through an intelligent network.

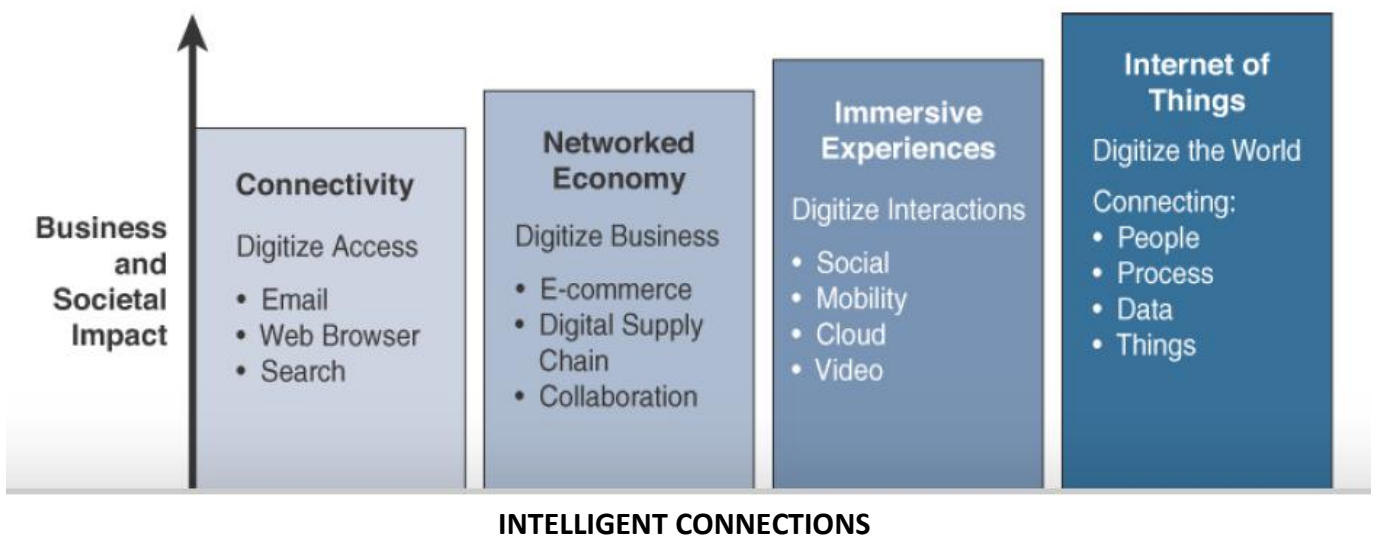
GOAL: The basic premise and goal of IoT is to “connect the unconnected.” This means that objects that are not currently joined to a computer network, namely the Internet, will be connected so that they can communicate and interact with people and other objects.

When objects and machines can be sensed and controlled remotely across a network, a tighter integration between the physical world and computers is enabled.

This allows for improvements in the areas of efficiency, accuracy, automation, and the enablement of advanced applications.

GENESIS OF IOT

The person credited with the creation of the term “Internet of Things” is Kevin Ashton. While working for Procter & Gamble in 1999, Kevin used this phrase to explain a new idea related to linking the company’s supply chain to the Internet.



The evolution of the Internet can be categorized into four phases. Each of these phases has had a profound impact on our society and our lives. These four phases are further defined in Table below.

Internet Phase	Definition
Connectivity (Digitize access)	This phase connected people to email, web services, and search so that information is easily accessed.
Networked Economy (Digitize business)	This phase enabled e-commerce and supply chain enhancements along with collaborative engagement to drive increased efficiency in business processes.
Immersive Experiences (Digitize interactions)	This phase extended the Internet experience to encompass widespread video and social media while always being connected through mobility. More and more applications are moved into the cloud.
Internet of Things (Digitize the world)	This phase is adding connectivity to objects and machines in the world around us to enable new services and experiences. It is connecting the unconnected.

IOT AND DIGITIZATION

IoT and *digitization* are terms that are often used interchangeably. In most contexts, this duality is fine, but there are key differences to be aware of.

At a high level, IoT focuses on connecting “things,” such as objects and machines, to a computer network, such as the Internet. IoT is a well-understood term used across the industry as a whole. On the other hand, digitization can mean different things to different people but generally encompasses the connection of “things” with the data they generate and the business insights that result.

Digitization, as defined in its simplest form, is the conversion of information into a digital format. Digitization has been happening in one form or another for several decades. For example, the whole photography industry has been digitized. Pretty much everyone has digital cameras these days, either standalone devices or built into their mobile phones. Almost no one buys film and takes it to a retailer to get it developed. The digitization of photography has completely changed our experience when it comes to capturing images.

CONVERGENCE OF IT AND OT

Until recently, information technology (IT) and operational technology (OT) have for the most part lived in separate worlds. 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. Typically, IT did not get involved with the production and logistics of OT environments.

Management of OT is tied to the lifeblood of a company. For example, if the network connecting the machines in a factory fails, the machines cannot function, and production may come to a standstill, negatively impacting business on the order of millions of dollars. On the other hand, if the email server (run by the IT department) fails for a few hours, it may irritate people, but it is unlikely to impact business at anywhere near the same level. **Table below 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

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.

IOT CHALLENGES

The most significant challenges and problems that IoT is currently facing are

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.
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.

IoT Network Architecture and Design

The unique challenges posed by IoT networks and how these challenges have driven new architectural models.

- Drivers Behind New Network Architectures
- Comparing IoT Architectures.
- A Simplified IoT Architecture
- The Core IoT Functional Stack
- IoT Data Management and Compute Stack

DRIVERS BEHIND NEW NETWORK ARCHITECTURES

This begins by comparing how using an architectural blueprint to construct a house is similar to the approach we take when designing a network. Take 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.

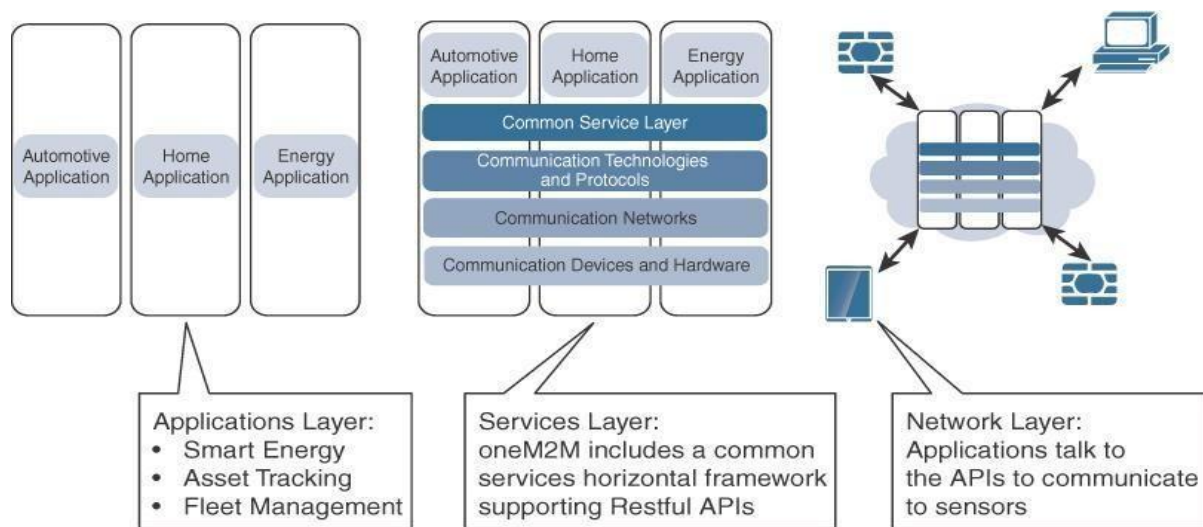
Challenge	Description	IoT Architectural Change Required
Scale	The massive scale of IoT endpoints (sensors) is far beyond that of typical IT networks.	The IPv4 address space has reached exhaustion and is unable to meet IoT's scalability requirements. Scale can be met only by using IPv6. IT networks continue 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.

COMPARING IOT ARCHITECTURES

The oneM2M IoT Standardized Architecture

In an effort to standardize the rapidly growing field of machine-to-machine (M2M) communications, the European Telecommunications Standards Institute (ETSI) created the M2M Technical Committee in 2008. The goal of this committee was to create a common architecture that would help accelerate the adoption of M2M applications and devices. Over time, the scope has expanded to include the Internet of Things.

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



The Main Elements of the oneM2M IoT Architecture

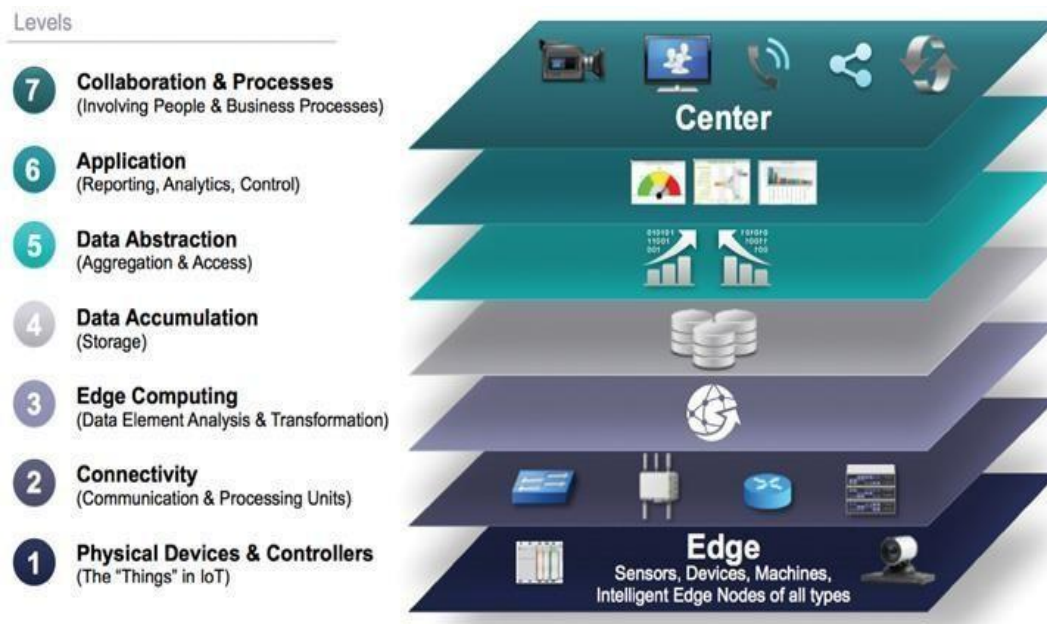
The oneM2M architecture divides IoT functions into three major domains: the application layer, the services layer, and the network layer

- **Applications layer:** The oneM2M architecture gives major attention to connectivity between devices and their applications. This domain 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:** This layer is shown as a horizontal framework across the vertical industry applications. At this layer, 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. Riding on top is the common services layer.
- **Network layer:** This is the communication domain for the IoT devices and endpoints. It includes the devices themselves and the communications network that links them. Embodiments of this communications infrastructure include wireless mesh technologies, such as IEEE 802.15.4, and wireless point-to-multipoint systems, such as IEEE 801.11ah.

The IoT World Forum (IoTWF) Standardized Architecture

This publish a seven-layer IoT architectural reference model.

- While various IoT reference models exist, the one put forth by the IoT World Forum offers a clean, simplified perspective on IoT and includes edge computing, data storage, and access. It provides a succinct way of visualizing IoT from a technical perspective. Each of the seven layers is broken down into specific functions, and security encompasses the entire model.



Using this reference model, we are able to achieve the following:

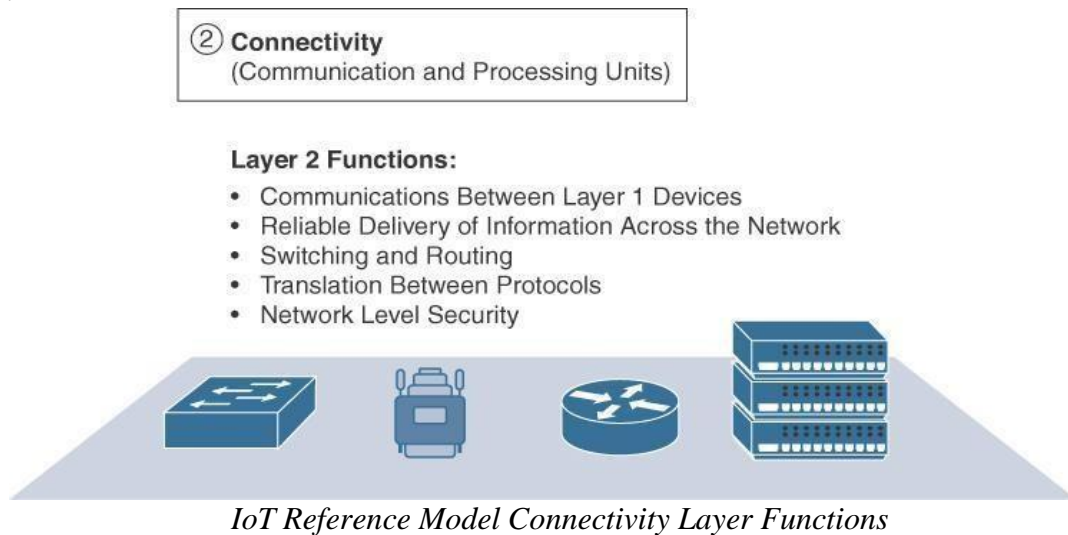
1. Decompose the IoT problem into smaller parts
2. Identify different technologies at each layer and how they relate to one another
3. Define a system in which different parts can be provided by different vendors
4. Have a process of defining interfaces that leads to interoperability
5. Define a tiered security model that is enforced at the transition points between levels

Layer 1: Physical Devices and Controllers Layer

The first layer of the IoT Reference Model is the 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 size of these “things” can range from almost microscopic sensors to giant machines in a factory. Their primary function is generating data and being capable of being queried and/or controlled over a network.

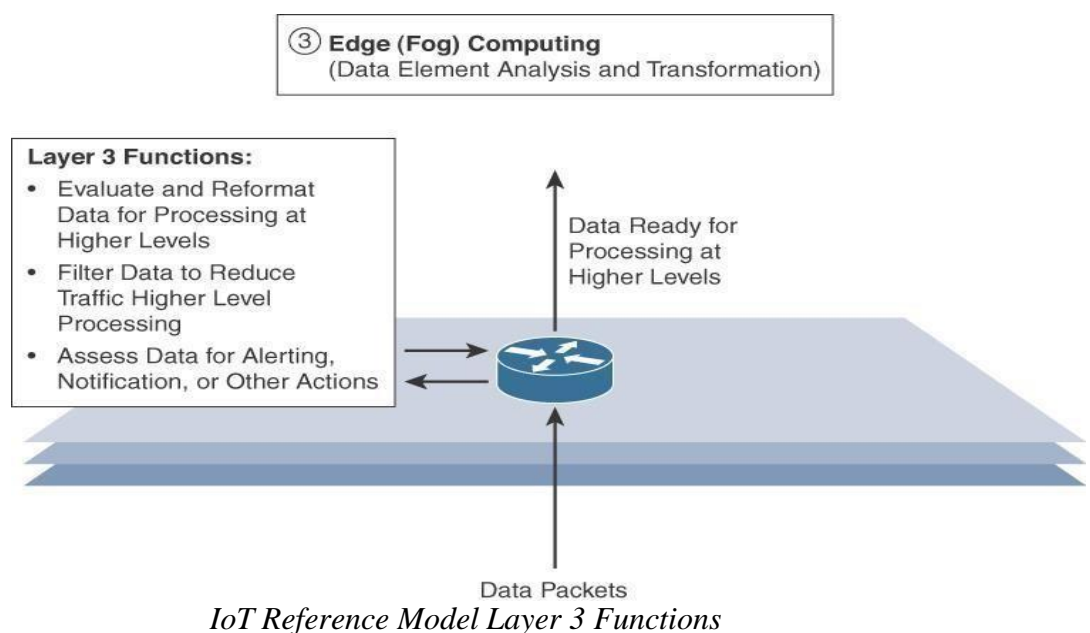
Layer 2: Connectivity Layer

In the second layer of the IoT Reference Model, the focus is on connectivity. The most important function of this IoT layer is the reliable and timely transmission of data. More specifically, this includes transmissions between Layer 1 devices and the network and between the network and information processing that occurs at Layer 3 (the edge computing layer).



Layer 3: Edge Computing Layer

Edge computing is the role of Layer 3. Edge computing is often referred to as the “fog” layer and is discussed in the section “Fog Computing,” later in this chapter. 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 and as close to the edge of the network as possible



Another important function that occurs at Layer 3 is the evaluation of data to see if it can be filtered or aggregated before being sent to a higher layer. This also allows for data to be reformatted or decoded, making additional processing by other systems easier. Thus, a critical function is assessing the data to see if predefined thresholds are crossed and any action or alerts need to be sent.

Upper Layers: Layers 4–7

The upper layers deal with handling and processing the IoT data generated by the bottom layer. For the sake of completeness, Layers 4–7 of the IoT Reference Model are summarized in [Table .](#)

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.

IT and OT Responsibilities in the IoT Reference Model

An interesting aspect of visualizing an IoT architecture this way is that you can start to organize responsibilities along IT and OT lines. Figure illustrates a natural demarcation point between IT and OT in the IoT Reference Model framework.

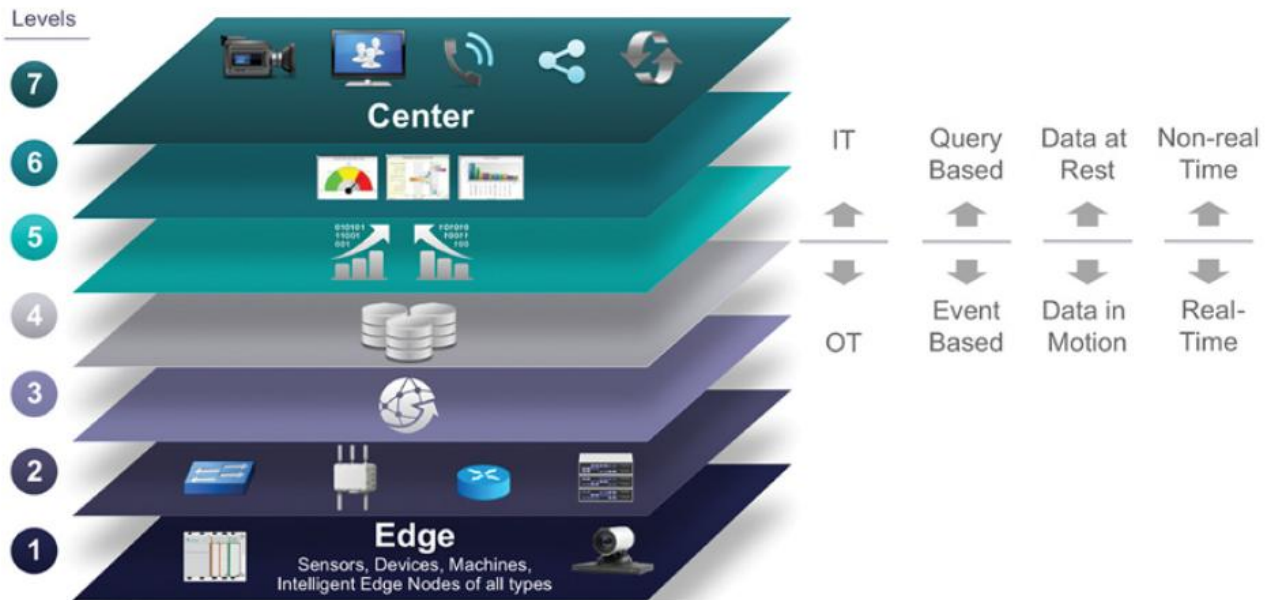


Figure: IoT Reference Model Separation of IT and OT

As demonstrated in Figure, IoT systems have to cross several boundaries beyond just the functional layers. 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. In the past, OT and IT have generally been very independent and had little need to even talk to each other. IoT is changing that paradigm. At the bottom, in the OT layers, the devices generate real-time data at their own rate—sometimes vast amounts on a daily basis. Not only does this result in a huge amount of data transiting the IoT network, but the sheer volume of data suggests that applications at the top layer will be able to ingest that much data at the rate required. To meet this requirement, data has to be buffered or stored at certain points within the IoT stack. Layering data management in this way throughout the stack helps the top four layers handle data at their own speed.

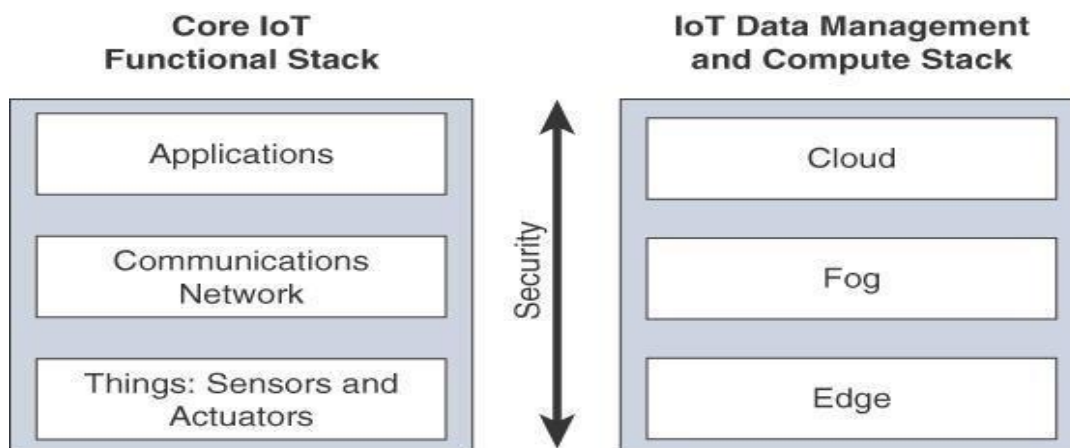
As a result, the real-time “data in motion” close to the edge has to be organized and stored so that it becomes “data at rest” for the applications in the IT tiers. The IT and OT organizations need to work together for overall data management.

Additional IoT Reference Models

In addition to the two IoT reference models already presented, several other reference models exist. These models are endorsed by various organizations and standards bodies and are often specific to certain industries or IoT applications. Table highlights these 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://vdvde-it.de/en .

A SIMPLIFIED IOT ARCHITECTURE



Simplified IoT Architecture

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:

- “Things” layer:
- Communications network layer
- Access network sublayer
- Gateways and backhaul network sublayer
- Network transport sublayer
- IoT network management sublayer
- Application and analytics layer

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

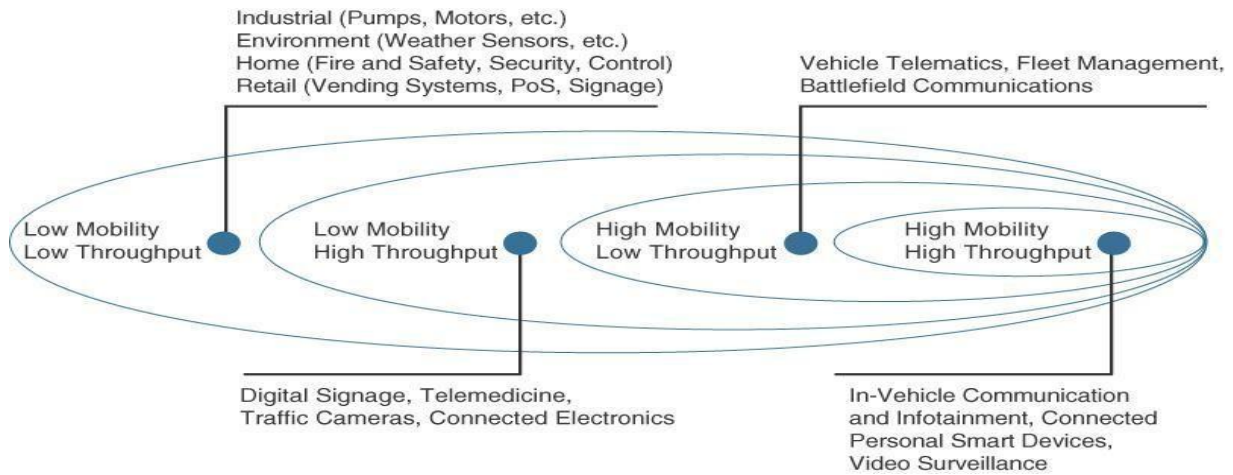
Layer 1: Things: Sensors and Actuators Layer

“Smart Objects: The ‘Things’ in IoT,” provides more in-depth information about smart objects. From an architectural standpoint, the variety of smart object types, shapes, and needs drive the variety of IoT protocols and architectures. One architectural classification could be:

- **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.
- **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 or because it is attached to a moving object.
- **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.

- **Simple or rich data:** This classification is based on the quantity of data exchanged at each report cycle
- **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.
- **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.

Below figure provides some examples of applications matching the combination of mobility and throughput requirements.



Example of Sensor Applications Based on Mobility and Throughput

Layer 2: Communications Network Layer

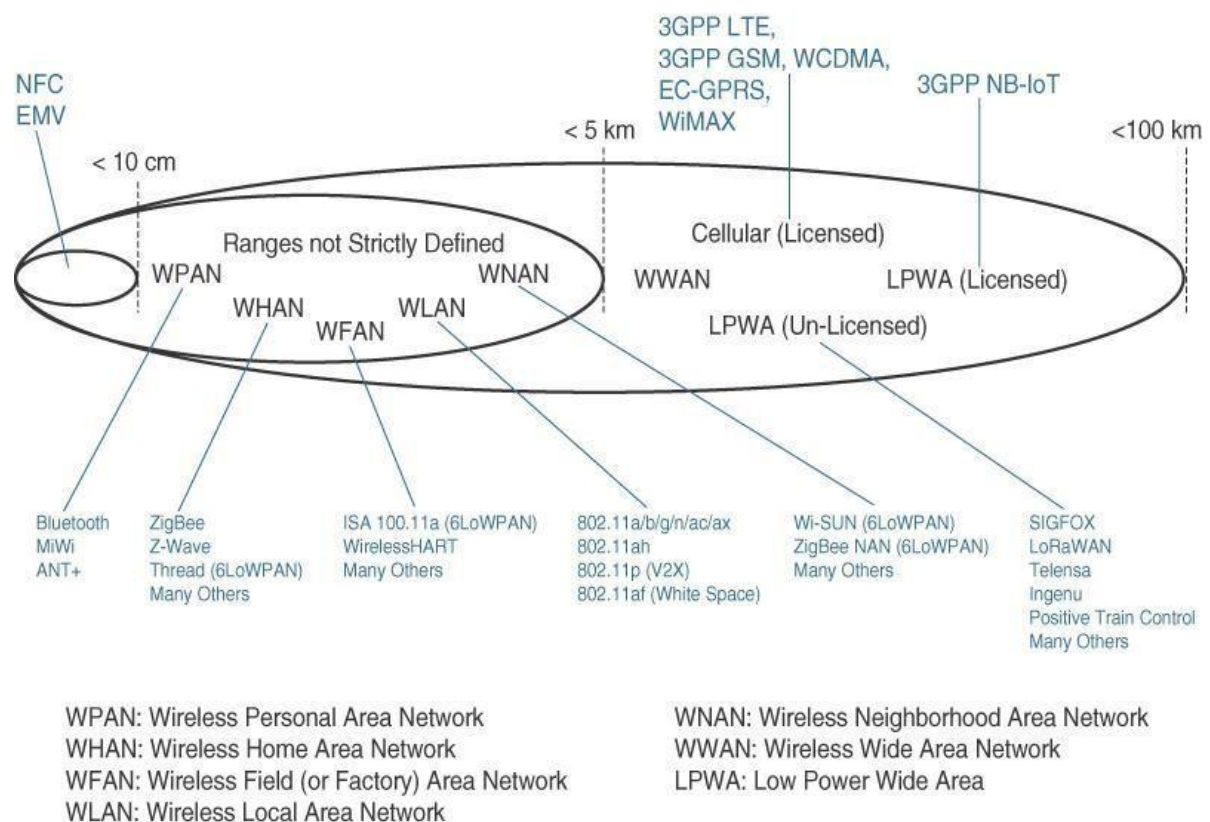
Once you have determined the influence of the smart object form factor over its transmission capabilities (transmission range, data volume and frequency, sensor density and mobility), you are ready to connect the object and communicate.

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.

Access Network Sublayer

There is a direct relationship between the IoT network technology you choose and the type of connectivity topology this technology allows. Each technology was designed with a certain number of use cases in mind (what to connect, where to connect, how much data to transport at what interval and over what distance). These use cases determined the frequency band that was expected to be most suitable, the frame structure matching the expected data pattern (packet size and communication intervals), and the possible topologies that these use cases illustrate.

One key parameter determining the choice of access technology is the range between the smart object and the information collector. Figure 2-9 lists some access technologies you may encounter in the IoT world and the expected transmission distances.



Access Technologies and Distances

- ✓ 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.

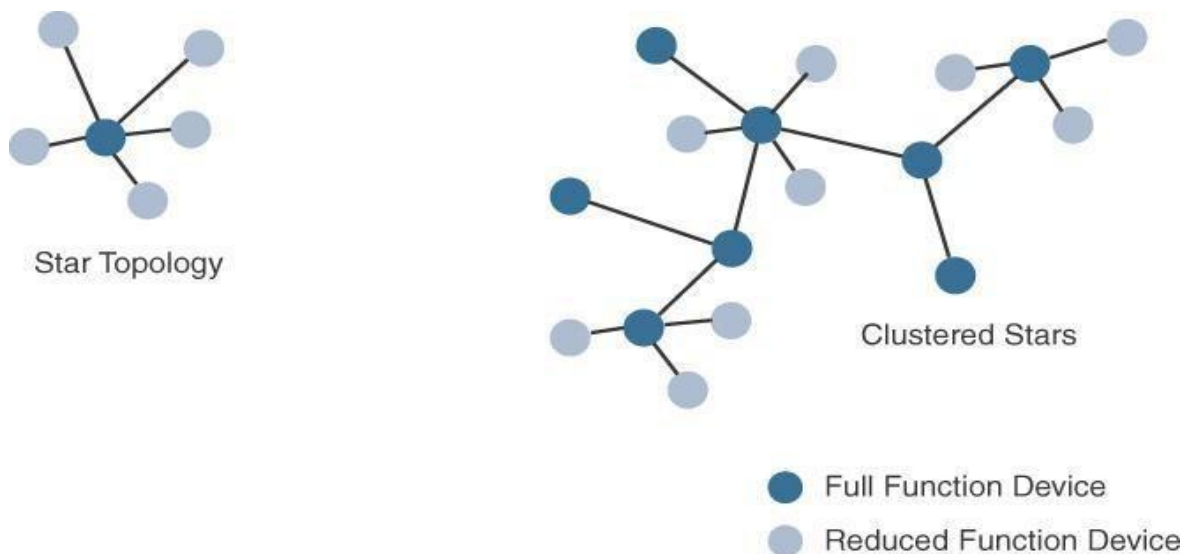
■ **FAN (field area network):** Scale of several tens of meters to several hundred meters. FAN typically refers to an outdoor area larger than a single group of house units. The FAN is often seen as “open space” (and therefore not secured and not controlled).

■ **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.

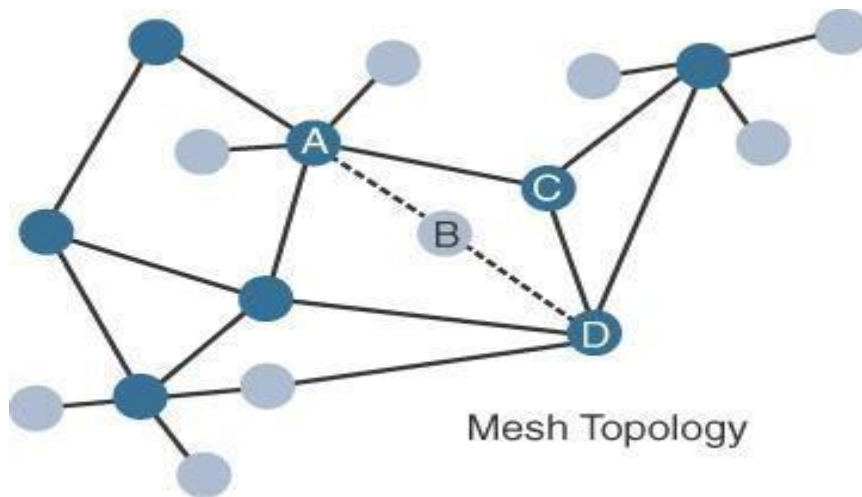
- ✓ Similar ranges also do not mean similar topologies. Some technologies offer flexible connectivity structure to extend communication possibilities:

■ **Point-to-point topologies**

■ **Point-to-multipoint**



Star and Clustered Star Topologies



Comparison of 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)

Architectural Considerations for WiMAX and Cellular Technologies

Layer 3: Applications and Analytics Layer

Once connected to a network, your smart objects exchange information with other systems. As soon as your IoT network spans more than a few sensors, the power of the Internet of Things appears in the applications that make use of the information exchanged with the smart objects.

Analytics Versus Control Applications

Multiple applications can help increase the efficiency of an IoT network. Each application collects data and provides a range of functions based on analyzing the collected data. It can be difficult to compare the features offered. From an architectural standpoint, one basic classification can be as follows:

■ **Analytics application:** This type of application collects data from multiple smart objects, processes the collected data, and displays information resulting from the data that was processed. The display can be about any aspect of the IoT network, from historical reports, statistics, or trends to individual system states. The important aspect is that the 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.

■ **Control application:** This type of application controls the behavior of the smart object or the behavior of an object related to the smart object. For example, a pressure sensor may be connected to a pump. A control application increases the pump speed when the connected sensor detects a drop in pressure. Control applications are very useful for controlling complex aspects of an IoT network with a logic that cannot be programmed inside a single IoT object, either because the configured changes are too complex to fit into the local system or because the configured changes rely on parameters that include elements outside the IoT object.

Data Versus Network Analytics

Analytics is a general term that describes processing information to make sense of collected data. In the world of IoT, a possible classification of the analytics function is as follows:

■ **Data analytics:** This type of analytics processes the data collected by smart objects and combines it to provide an intelligent view related to the IoT system. At a very basic level, a dashboard can display an alarm when a weight sensor detects that a shelf is empty in a store. In a more complex case, temperature, pressure, wind, humidity, and light levels collected from thousands of sensors may be combined and then processed to determine the likelihood of a storm and its possible path .

■ **Network analytics:** Most IoT systems are built around smart objects connected to the network. A loss or degradation in connectivity is likely to affect the efficiency of the system. Such a loss can have dramatic effects. For example, open mines use wireless networks to automatically pilot dump trucks. A lasting loss of connectivity may result in an accident or degradation of operations efficiency (automated dump trucks typically stop upon connectivity loss). On a more minor scale, loss of connectivity means that data stops being fed to your data analytics platform, and the system stops making intelligent analyses of the IoT system.

Data Analytics Versus Business Benefits

Data analytics is undoubtedly a field where the value of IoT is booming. Almost any object can be connected, and multiple types of sensors can be installed on a given object. Collecting and interpreting the data generated by these devices is where the value of IoT is realized.

Smart Services

- The ability to use IoT to improve operations is often termed “smart services.” This term is generic, and in many cases the term is used but its meaning is often stretched to include one form of service or another where an additional level of intelligence is provided.

- Smart services can also be used to measure the efficiency of machines by detecting machine output, speed, or other forms of usage evaluation.
- 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.

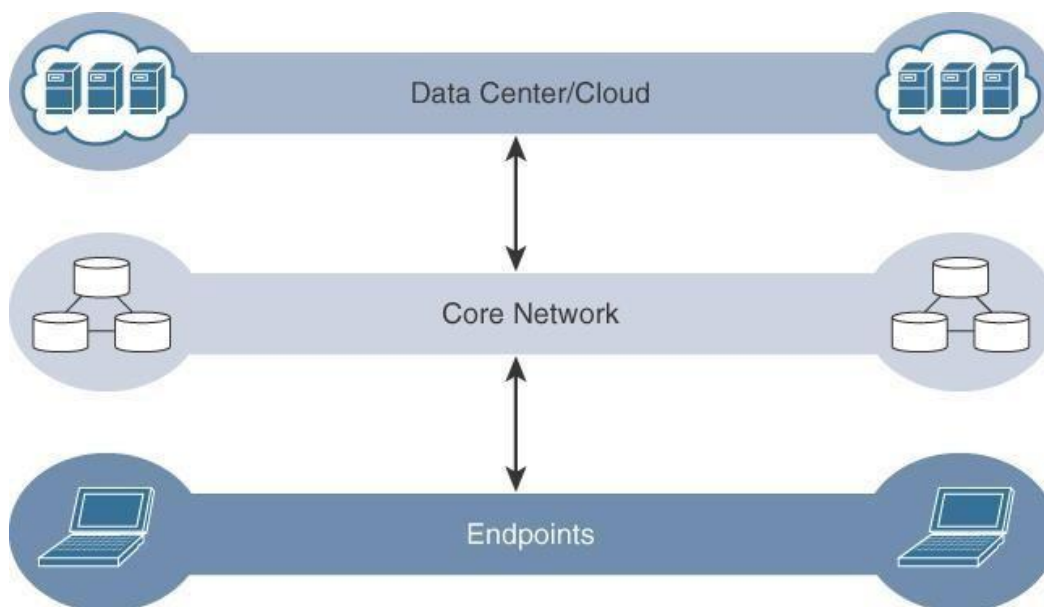
IOT DATA MANAGEMENT AND COMPUTE STACK

This model also has limitations. As data volume, the variety of objects connecting to the network, and the need for more efficiency increase, new requirements appear, and those requirements tend to bring the need for data analysis closer to the IoT system. These new requirements include the following:

■ **Minimizing latency:** Milliseconds matter for many types of industrial systems, such as when you are trying to prevent manufacturing line shutdowns or restore electrical service. 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:** Offshore oil rigs generate 500 GB of data weekly. Commercial jets generate 10 TB for every 30 minutes of flight. 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 environmental conditions in one area will trigger a local response independent from the conditions of another site hundreds of miles away. Analyzing both areas in the same cloud system may not be necessary for immediate efficiency.



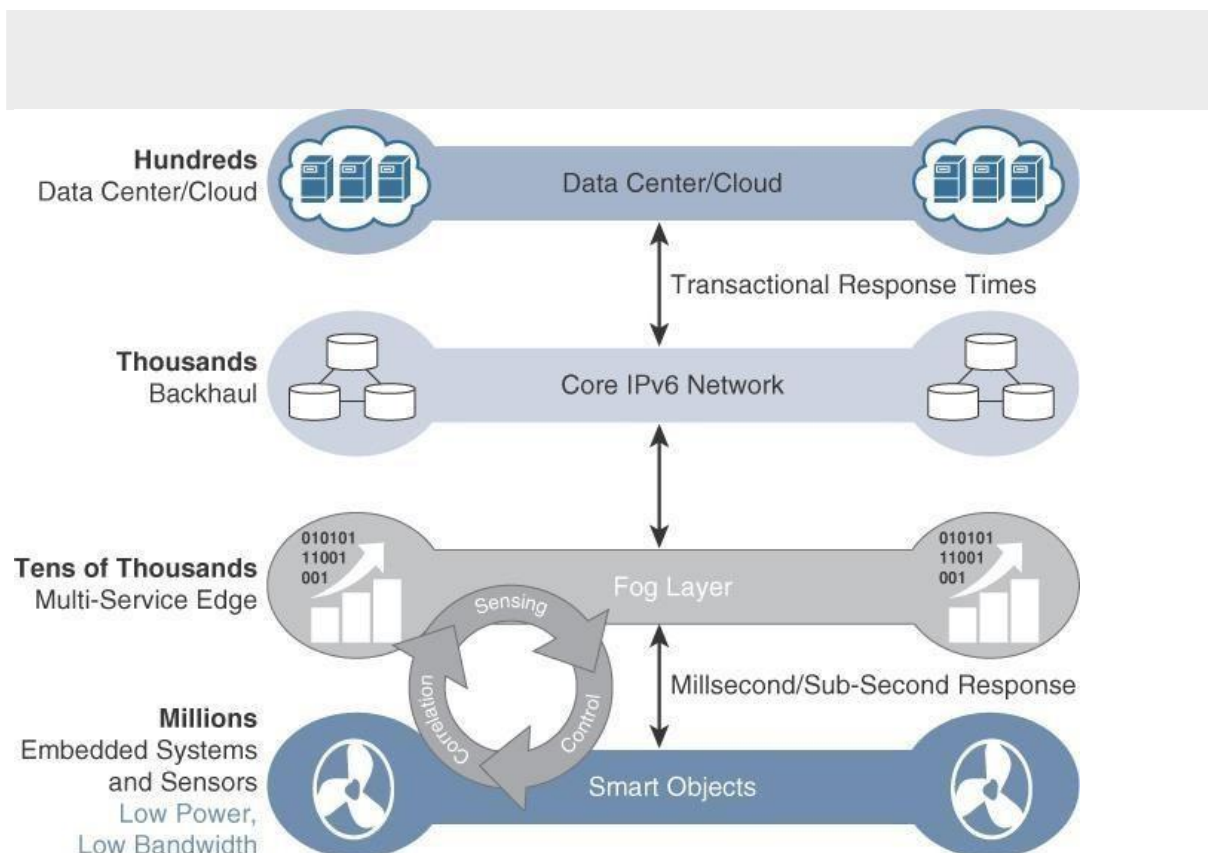
The Traditional IT Cloud Computing Model

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

- 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.
- 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.
- 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.
- 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).
- 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.



The IoT Data Management and Compute Stack with Fog Computing

Fog services are typically accomplished very close to the edge device, sitting as close to the IoT endpoints as possible. One significant advantage of this is that the fog node has contextual awareness of the sensors it is managing because of its geographic proximity to those sensors. For example, there might be a fog router on an oil derrick that is monitoring all the sensor activity at that location. Because the fog node is able to analyze information from all the sensors on that derrick, it 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. In this way, it is performing distributed analytics such that the volume of data sent upstream is greatly reduced and is much more useful to application and analytics servers residing in the cloud.

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.

The defining characteristic of fog computing are as follows:

- **Contextual location awareness and low latency:** The fog node sits as close to the IoT endpoint as possible to deliver distributed computing.
- **Geographic distribution:** In sharp contrast to the more centralized cloud, the services and applications targeted by the fog nodes demand widely distributed deployments.
- **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.
- **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.
- **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

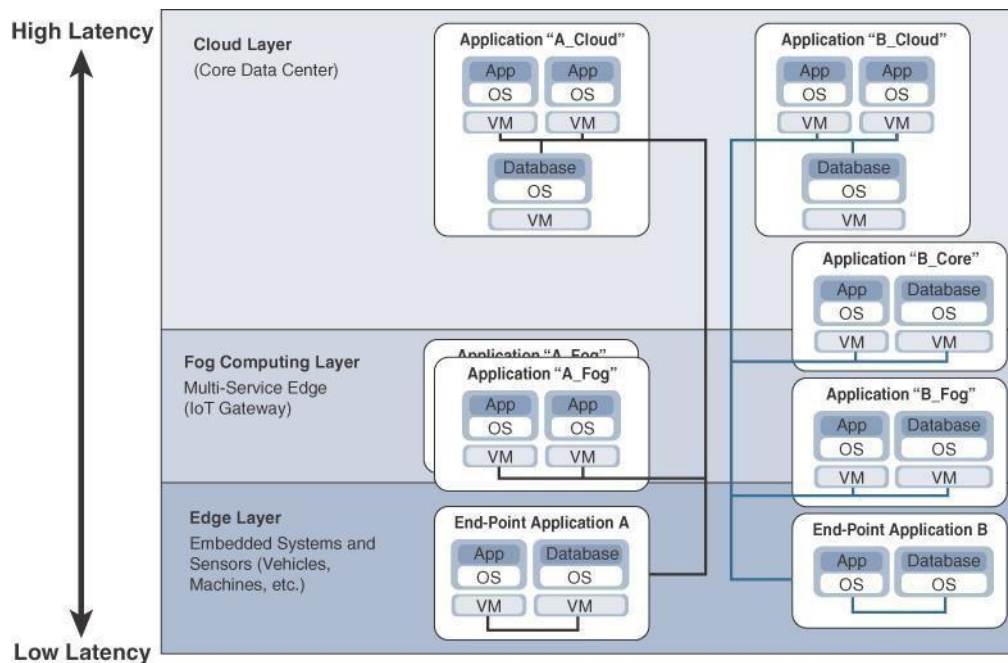
Note

Edge computing is also sometimes called “mist” computing. If clouds exist in the sky, and fog sits near the ground, then mist is what actually sits on the ground. Thus, the concept of mist is to extend fog to the furthest point possible, right into the IoT endpoint device itself.

The Hierarchy of Edge, Fog, and Cloud

It is important to stress that edge or fog computing in no way replaces the cloud. Rather, they complement each other, and many use cases actually require strong cooperation between layers. In the same way that lower courts do not replace the supreme court of a

country, edge and fog computing layers simply act as a first line of defense for filtering, analyzing, and otherwise managing data endpoints. This saves the cloud from being queried by each and every node for each event.



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.

In summary, when architecting an IoT network, you should consider the amount of data to be analyzed and the time sensitivity of this data. Understanding these factors will help you decide whether cloud computing is enough or whether edge or fog computing would improve your system efficiency. Fog computing accelerates awareness and response to events by eliminating a round trip to the cloud for analysis. It avoids the need for costly bandwidth additions by offloading gigabytes of network traffic from the core network. It also protects sensitive IoT data by analyzing it inside company walls.

MODULE-2

SMART OBJECTS

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.

Some of the fundamental building blocks of IoT networks are

- Sensors
- Actuators
- Smart Objects

Sensors:

- A sensor does exactly as its name indicates: It senses.
- 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.
- Sensors are not limited to human-like sensory data.
- They are able to provide an extremely wide spectrum of rich and diverse measurement data with far greater precision than human senses.
- Sensors provide superhuman sensory capabilities.
- Sensors can be readily embedded in any physical objects that are easily connected to the Internet by wired or wireless networks, they can interpret their environment and make intelligent decisions.

Sensors have been grouped into different categories

- **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.

The physical phenomenon a sensor is measuring is shown in Table-2.1

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

- A fascinating use case to highlight the power of sensors and IoT is in the area of precision agriculture (sometimes referred to as 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.

Different types of sensors in a smart phone is shown in figure 2.1



Figure 2.1: Sensors in a smart phone

Actuators:

- Actuators are natural complements to sensors.
- Figure 2.2 demonstrates the symmetry and complementary nature of these two types of devices.
- Sensors are designed to sense and measure practically any measurable variable in the physical world.
- They convert their measurements (typically analog) into electric signals or digital representations that can be consumed by an intelligent agent (a device or a human).
- Actuators, on the others hand, 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.

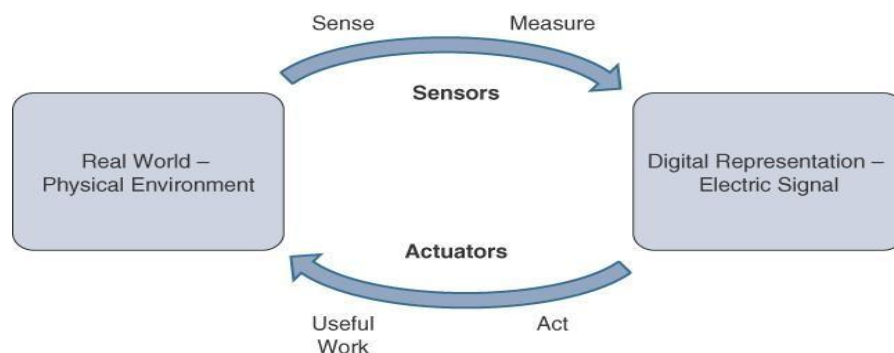


Figure 2.2 : How Sensors and Actuators Interact with the Physical World

Much like sensors, actuators also 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.

Different types of Actuators are presented in Table -2.2

Type	Examples
Mechanical actuators	Lever, screw jack, hand crank
Electrical actuators	Thyristor, bipolar 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, magnetostrictive material, bimetallic strip, piezoelectric bimorph
Micro- and nanoactuators	Electrostatic motor, microvalve, comb drive

Table -

2.2: Actuator Classification by Energy Type

Micro-Electro-Mechanical Systems (MEMS)

- Micro-electro-mechanical systems (MEMS referred to as micro-machines, can integrate and combine electric and mechanical elements, such as sensors and actuators, on a very small (millimeter or less) scale.
- The combination of tiny size, low cost, and the ability to mass produce makes MEMS an attractive option for a huge number of IoT applications.

Ex: Inkjet printers use micropump MEMS. Smart phones also use MEMS technologies for things like accelerometers and gyroscopes

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. A *smart object*, is a device that has, at a minimum, the following four defining characteristics

- **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.
- **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.
- **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.

Trends in Smart Objects:

The broad generalizations and trends impacting IoT are

- **Size is decreasing:** 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. Some battery-powered sensors last 10 or more years without battery replacement.
- **Processing power is increasing:** Processors are continually getting more powerful and smaller.
- **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

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.
- 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.

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

Wireless Sensor Networks (WSNs)

Wireless sensor networks are made up of wirelessly connected smart objects, which are sometimes referred to as *motest*. 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. Figure 2.3 below 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.

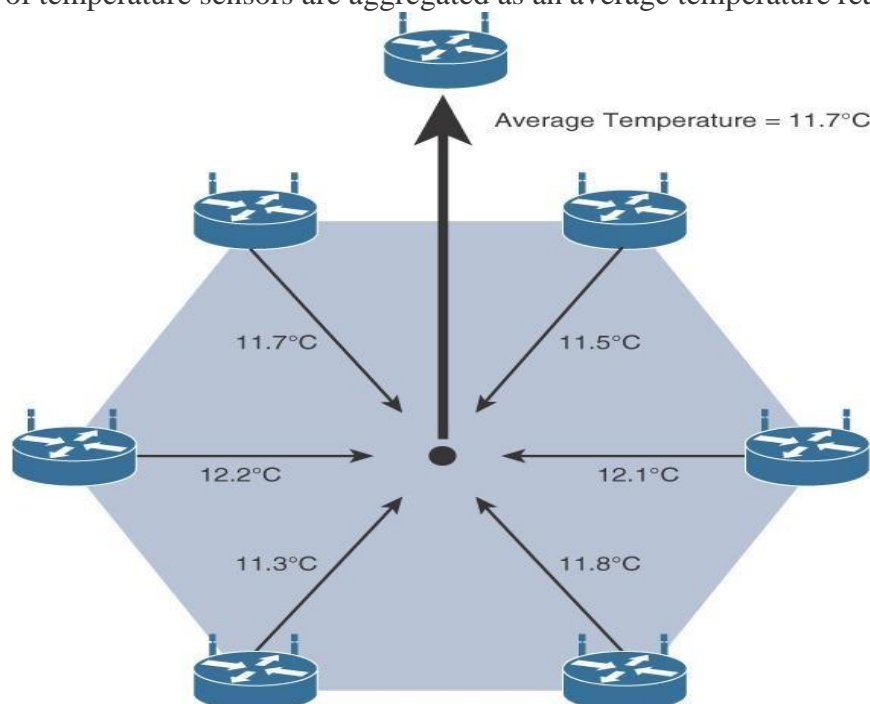


Figure 2.3 Data Aggregation in Wireless Sensor Networks

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. 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.

Communication Protocols for Wireless Sensor Networks:

- Any communication protocol must be able to scale to a large number of nodes.
- Likewise, when selecting a communication protocol, you must carefully take into account the requirements of the specific application.
- Also 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.
- 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.
- IoT is one of those rare technologies that impacts all verticals and industries, which means standardization of communication protocols is a complicated task, requiring protocol definition across multiple layers of the stack, as well as a great deal of coordination across multiple standards development organizations.

Connecting smart objects

The characteristics and attributes considered when selecting and dealing with connecting smart objects are

1) Range: It defines how far does the signal need to be propagated? That is, what will be the area of coverage for a selected wireless technology? The below figure 2.4 shows the range considered

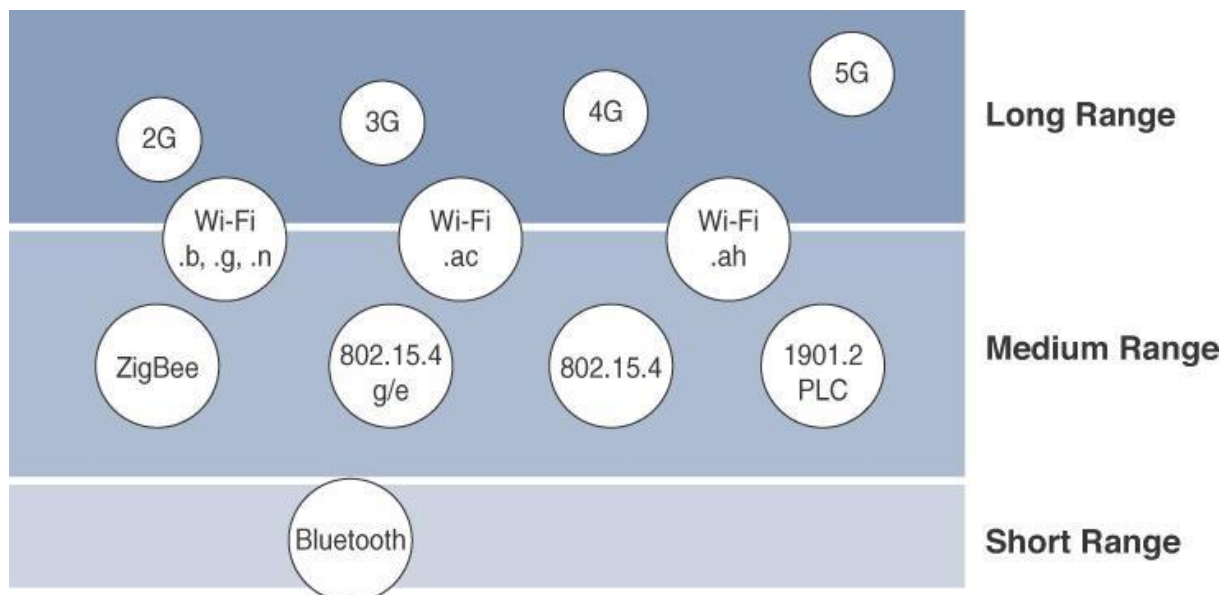


Figure 2.4 Wireless Access Landscape

- **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.
- **Medium Range:**
 - 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.
 - 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.

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.
- 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
- Unlicensed spectrum is usually simpler to deploy than licensed because it does not require a service provider.
- Some communications within the ISM bands operate in the sub-GHz range.
- Sub-GHz bands are used by protocols such as IEEE 802.15.4, 802.15.4g, and 802.11ah, and LPWA technologies such as LoRa and Sigfox.
- The most well-known ranges are centered on 169 MHz, 433 MHz, 868 MHz, and 915 MHz.
- The 868 MHz band is applicable to IoT access technologies such as IEEE 802.15.4 and 802.15.4g, 802.11ah, and LoRaWAN.

Power Consumption:

- Battery-powered nodes bring much more flexibility to IoT devices.
- These nodes are often classified by the required lifetimes of their batteries.
- A powered node has a direct connection to a power source, and communications are usually not limited by power consumption criteria.
- 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).

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.
- Peer-to-peer topologies allow any device to communicate with any other device as long as they are in range of each other.
- Peer-to-peer topologies enable more complex formations, such as a mesh networking topology.

The figure 2.5 below represents the various topology.

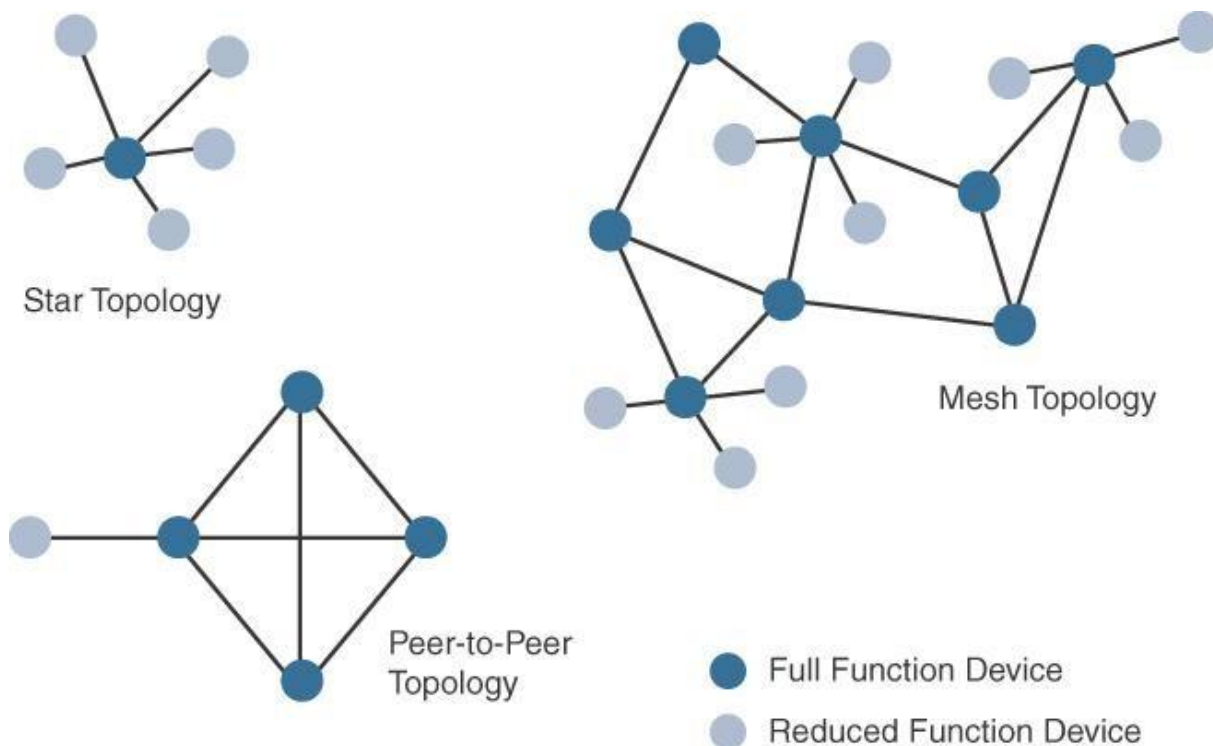


Figure 2.5 Star, Peer-to-Peer, and Mesh Topologies

- The disadvantage of sub-GHz frequency bands is their lower rate of data delivery compared to higher frequencies.
- Example: Indoor Wi-Fi deployments are mostly a set of nodes forming a star topology around their access points (APs).
- Outdoor Wi-Fi may consist of a mesh topology for the backbone of APs, with nodes connecting to the APs in a star topology.
- IEEE 802.15.4 and 802.15.4g and even wired IEEE 1901.2a PLC are generally deployed as a mesh topology.
- 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.

Constrained Devices:

Constrained nodes have limited resources that impact their networking feature set and capabilities. Constrained nodes can be broken down into different classes such as shown in Table 2.3:

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 2.3 Classes of Constrained Nodes, as Defined by RFC 7228

- Constrained-node networks are often referred to as low-power and lossy networks (LLNs).
- 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.
- 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.
- 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.
- Short-range technologies can also provide medium to high data rates that have enough throughput to connect a few endpoints.
- 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
- When considering constrained access network technologies, it is important to review the MAC payload size characteristics required by applications.
- In addition, you should be aware of any requirements for IP.

- The minimum IPv6 MTU size is expected to be 1280 bytes. Therefore, the fragmentation of the IPv6 payload has to be considered by link layer access protocols with smaller MTUs.
- 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.

IoT Access Technologies

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.
- This access technology enables easy installation using a compact protocol stack while remaining both simple and flexible.
- 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.
- Interference and multipath fading occur with IEEE 802.15.4 because it lacks a frequency-hopping technique.

❖ 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).
- 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.

Some of the most well-known protocol stacks based on 802.15.4 are as shown in Table 2.4

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

Protocol Stacks Utilizing IEEE 802.15.4

➤ **ZigBee:**

- It is an IoT solution for interconnecting smart objects.
- ZigBee solutions are aimed at smart objects and sensors that have low bandwidth and low power needs.
- The Zigbee specification has undergone several revisions.
- In the 2006 revision, sets of commands and message types were introduced, and increased in number in the 2007 (called Zigbee pro) iteration, to achieve different functions for a device, such as metering, temperature, or lighting control.
- These sets of commands and message types are called clusters.
- Ultimately, these clusters from different functional domains or libraries form the building blocks of Zigbee application profiles.
- Vendors implementing pre-defined Zigbee application profiles like Home Automation or Smart Energy can ensure interoperability between their products.
- 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 the below figure 2.6.

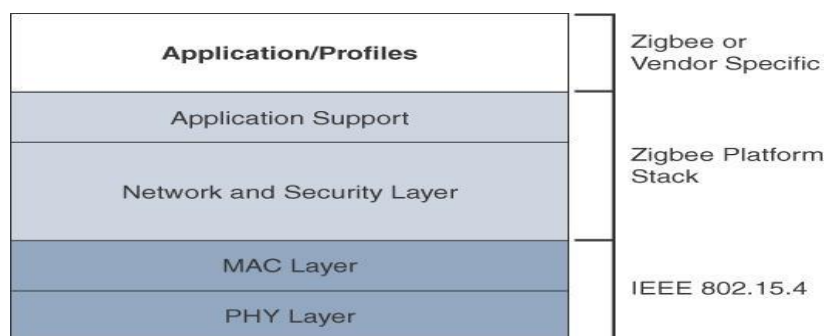


Figure 2.6 High-Level ZigBee Protocol Stack

- 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 is one of the most well-known protocols built on an IEEE 802.15.4 foundation. On top of the 802.15.4 PHY and MAC layers, ZigBee specifies its own network and security layer and application profiles.

➤ **ZigBee IP**

- 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. Any other applications that need a standards-based IoT stack can utilize Zigbee IP. The ZigBee IP stack is shown in below figure 2.7.

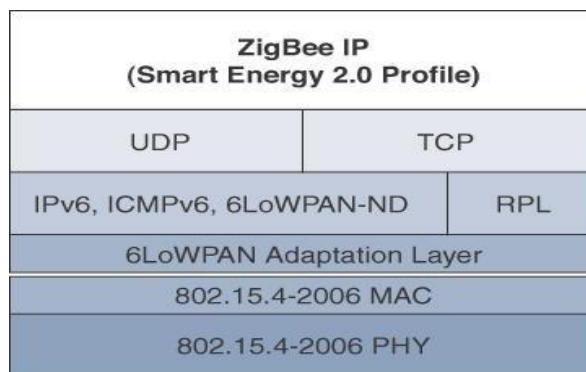


Figure 2.7 ZigBee IP Protocol Stack

- ZigBee IP supports 6LoWPAN as an adaptation layer.
- 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.

❖ **802.15.4 Physical and MAC 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.
- The 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
- IEEE 802.15.4-2006, 802.15.4-2011, and IEEE 802.15.4-2015 introduced additional PHY communication options, including the following:
 - **OQPSK PHY:** 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.

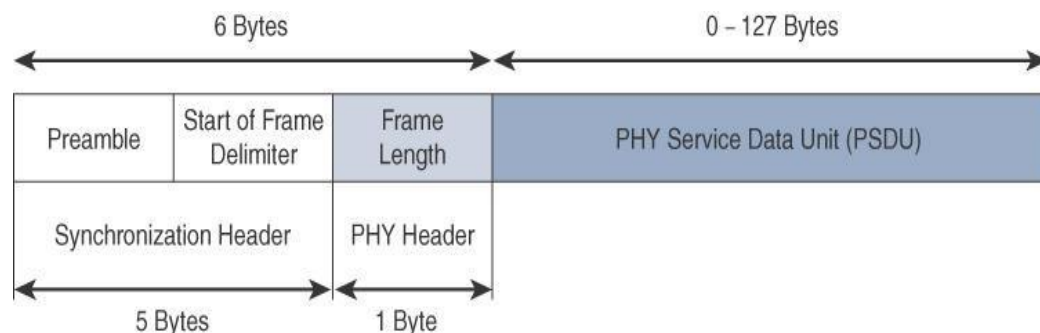
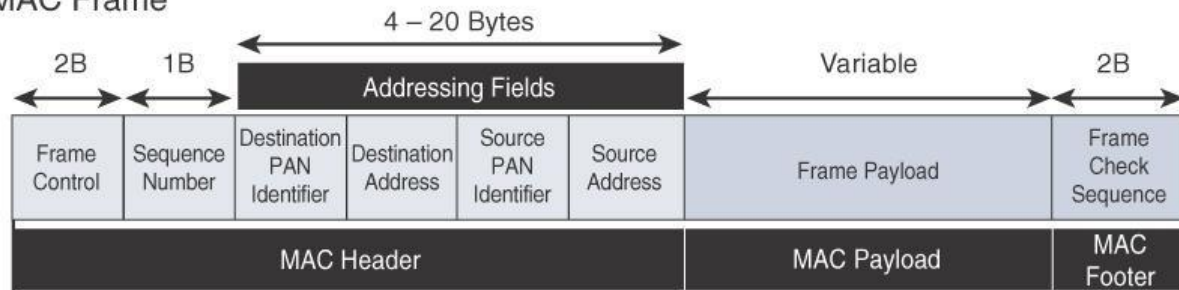


Figure 2.8 IEEE 802.15.4 PHY Format

- The PHY Header portion of the PHY frame is shown in Figure 2.8 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.
- 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 2.9. In Figure 2.9, 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.

MAC Frame



PHY Frame

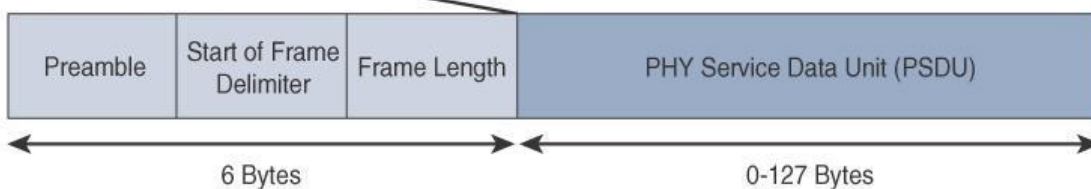


Figure 2.9 IEEE 802.15.4 MAC Format

- 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.
- The MAC Payload field varies by individual frame type.
- 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.
- 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 2.10 shows an example of an 802.15.4 mesh network with a PAN ID of 1.

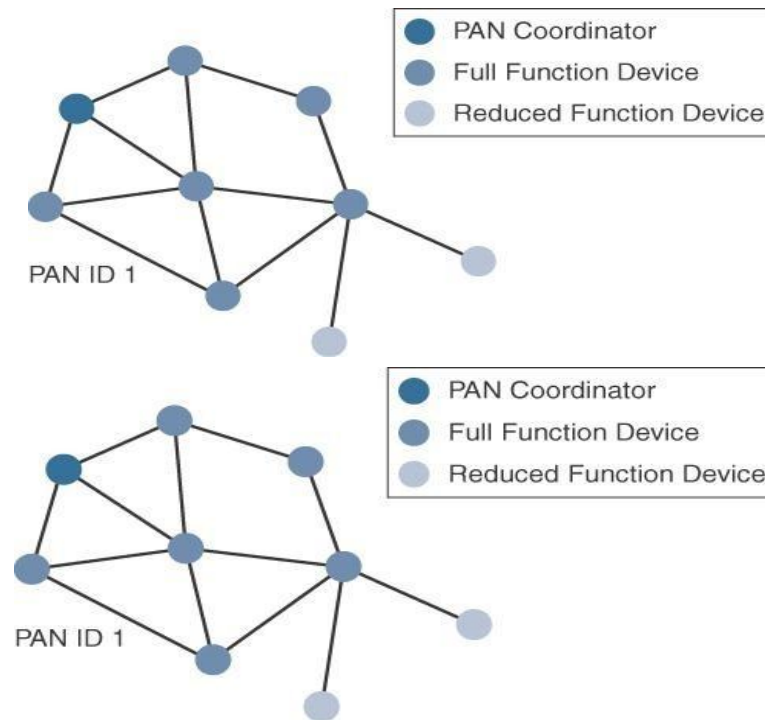
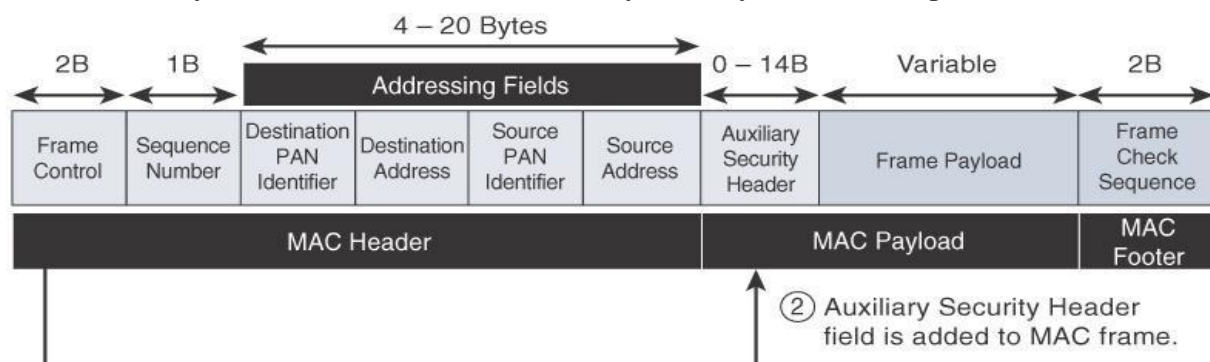


Figure 2.10: 802.15.4 Sample Mesh Network Topology

- FFD (full-function devices) acts as a PAN coordinator to deliver services that allow other devices to associate and form a cell or PAN.
- 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.
- 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.
- The figure 2.11 below 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.



① Security Enabled bit in Frame Control is set to 1.

② Auxiliary Security Header field is added to MAC frame.

Figure 2.11: Frame Format with the Auxiliary Security Header Field for 802.15.4-2006 and Later Versions

IEEE 802.15.4g and 802.15.4e

- 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.
- 802.15.4g seeks to optimize large outdoor wireless mesh networks for field area networks (FANs)
- 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
 - Environmental wireless sensors in smart cities
 - Electrical vehicle charging stations
 - Smart parking meters
 - Microgrids
 - Renewable energy.

❖ Standardization and Alliances:

- 802.15.4g-2012 and 802.15.4e-2012 are simply amendments to IEEE 802.15.4-2011.
- Same IEEE 802.15 Task Group 4 standards body authors, maintains, and integrates them into the next release of the core specification.
- To guarantee interoperability, the Wi-SUN Alliance was formed.
- It defines communication profiles for smart utility and related networks.
- These profiles are based on open standards, such as 802.15.4g-2012, 802.15.4e-2012, IPv6, 6LoWPAN, and UDP for the FAN profile.
- The Wi-SUN Alliance performs the same function as the Wi-Fi Alliance and WiMAX Forum

❖ 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.\
- 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.

❖ MAC Layer:

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.
 - A number of time slots are defined as a “slot frame,” which is regularly repeated to provide “guaranteed access.”
 - The transmitter and receiver agree on the channels and the timing for switching between channels through the combination of a global time slot counter and a global channel hopping sequence list, as computed on each node to determine the channel of each time slot.
 - TSCH adds robustness in noisy environments and smoother coexistence with other wireless technologies, especially for industrial use cases.
- **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 2.11 details this format

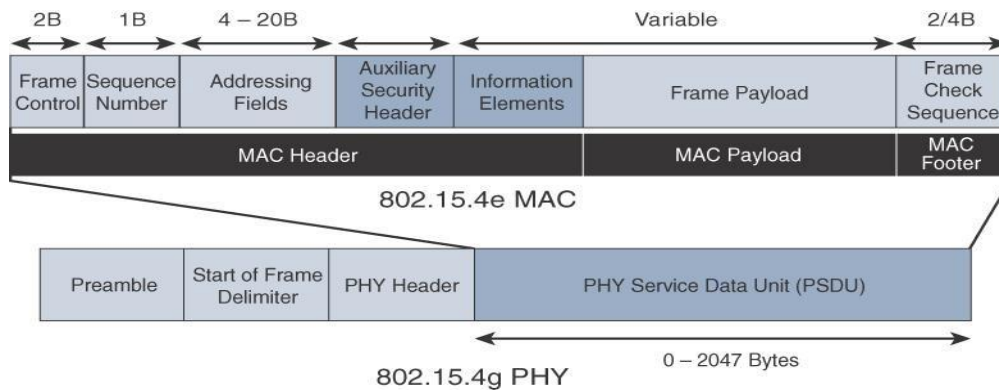


Figure 2.11: IEEE 802.15.4g/e MAC Frame Format

❖ **Topology:**

- Deployments of IEEE 802.15.4g-2012 are mostly based on a mesh topology.
- 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.
- Support for battery-powered nodes with a long lifecycle requires optimized Layer 2 forwarding or Layer 3 routing protocol implementations.
- This provides an extra level of complexity but is necessary in order to cope with sleeping battery-powered nodes.

❖ **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 2.12 shows a high-level overview of the security associated with an IEEE 802.15.4e MAC frame.

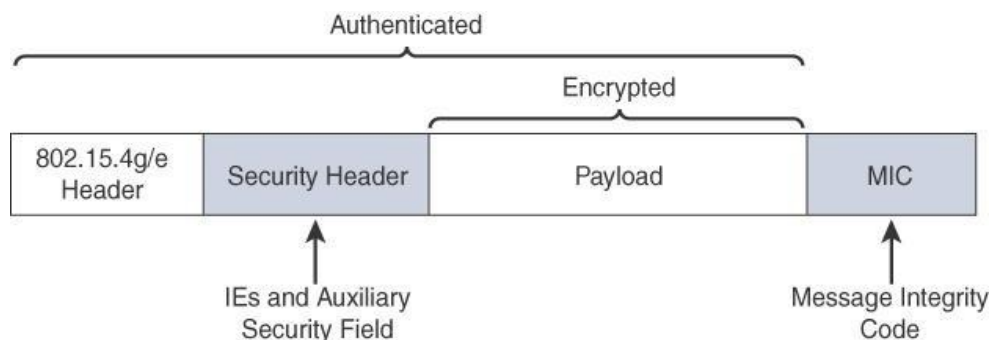


Figure 2.12: IEEE 802.15.4g/e MAC Layer Security

- The MIC is a unique value that is calculated based on the frame contents.
- The Security Header field denoted in Figure 2.12 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.

IEEE 1901.2a

- 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 leverages 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.

❖ Standardization and Alliances

- The IEEE 1901.2 working group published the IEEE 1901.2a specification in November 2013.
- IEEE 1901.2 working group only looked at standardizing the NB-PLC PHY and MAC layers independently of the upper layers.
- Using the 802.15.4e Information Element fields eases support for IEEE 802.15.9 key management.
- 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.

❖ Physical Layer

- NB-PLC is defined for frequency bands from 3 to 500 kHz.
- Figure 2.13 shows the various frequency bands for NB-PLC. The most well-known bands are regulated by CENELEC (Comité Européen de Normalisation Électro technique) and the FCC (Federal Communications Commission).
- The two ARIB frequency bands are ARIB 1, 37.5–117.1875 kHz, and ARIB 2, 154.6875–403.125 kHz.

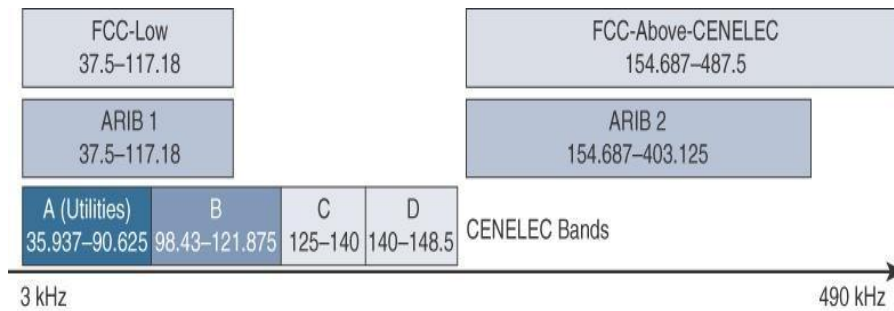


Figure 2.13 NB-PLC Frequency Bands

- With IEEE 1901.2a, the data throughput rate has the ability to dynamically change, depending on the modulation type and tone map.
- 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.

❖ **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.
- One of the key components brought from 802.15.4e to IEEE 1901.2a is information elements.
- Figure 2.14 provides an overview of the general MAC frame format for IEEE 1901.2.

3	2	1	0/2	0/2/8	0/2	0/2/8	0/5/6/ 10/14	Variable	Variable	2
Segment Control	Frame Control	Seq. Number	Dest. PAN Identifier	Dest. Address	Source PAN Identifier	Source Address	Auxiliary Security Header	Information Elements	Frame Payload	Frame Check Sequence
MAC Header (MHR)								MAC Protocol Data Unit (MPDU) Payload		MAC Footer (MFR)

Figure 2.14: General MAC Frame Format for IEEE 1901.2

- IEEE 1901.2 has a Segment Control field.
- 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.

❖ **Security:**

- IEEE 1901.2a security offers similar features to IEEE 802.15.4g. Encryption and authentication are performed using AES. I
- In addition, IEEE 1901.2a aligns with 802.15.4g in its ability to support the IEEE 802.15.9 Key Management Protocol.
- 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:**

- G3-PLC (now ITU G.9903)
- 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 802.11ah**

- In unconstrained networks, IEEE 802.11 Wi-Fi is certainly the most successfully deployed wireless technology.
- 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.
- Hence 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

- In July 2010, the IEEE 802.11 working group decided to work on an “industrial Wi-Fi” and created the IEEE 802.11ah group.
- The 802.11ah specification would operate in unlicensed sub-GHz frequency bands, similar to IEEE 802.15.4 and other LPWA technologies.
- For the 802.11ah standard, the Wi-Fi Alliance defined a new brand called Wi-Fi HaLow.
- It is similar to the word “hello” but it is pronounced “hay-low.”

❖ Physical Layer

- IEEE 802.11ah essentially provides an additional 802.11 physical layer operating in unlicensed sub-GHz bands.
- 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.
- Ex: 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.
 - **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.
 - **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.

❖ Topology

- While IEEE 802.11ah is deployed as a star topology, it includes a simple hops relay operation to extend its range.
- 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.
- The transmit rate reduces as you move further from the access point via relay clients.
- 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 neighbouring access points is problematic.
- Figure 2.15 shows an example of 802.11ah sectorization.

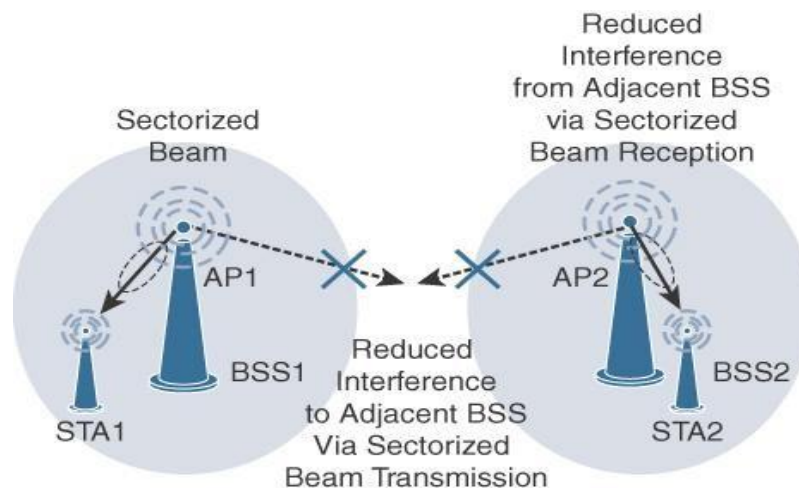


Figure 2.15 :IEEE 802.11ah Sectorization

❖ Security

- Similar to IEEE 802.11 specifications

❖ Competitive Technologies

- Competitive technologies to IEEE 802.11ah are IEEE 802.15.4 and IEEE 802.15.4e



LoRaWAN:

- It is an unlicensed-band LPWA(Low-Power Wide-Area) technology.

❖ **Standardization and Alliances**

- 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.
- The LoRa Alliance quickly achieved industry support and currently has hundreds of members.
- LoRa Alliance uses the term LoRaWAN to refer to its architecture and its specifications that describe end-to-end LoRaWAN communications and protocols.
- Figure 2.16 provides a high-level overview of the LoRaWAN layers.

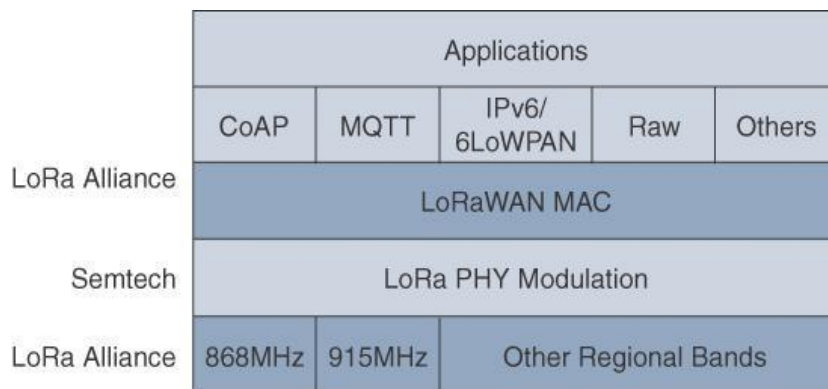


Figure 2.16 LoRaWAN Layers

❖ **Physical Layer**

- 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.
- A LoRa gateway is deployed as the center hub of a star network architecture.
- It uses multiple transceivers and channels and can demodulate multiple channels at once or even demodulate multiple signals on the same channel simultaneously.
- LoRa gateways serve as a transparent bridge relaying data between endpoints, and the endpoints use a single-hop wireless connection to communicate with one or many gateways.
- 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.
- 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.

❖ **MAC Layer**

- 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 2.17 shows a high-level LoRaWAN MAC frame format.

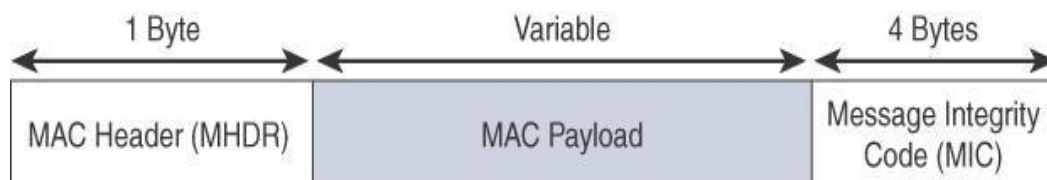


Figure 2.17: High-Level LoRaWAN MAC Frame Format

- In version 1.0.x, LoRaWAN utilizes six MAC message types
 - **Join request** : over-the-air (OTA) activation and joining the network.
 - **Join accept messages**: over-the-air (OTA) activation and joining the network.
 - **Unconfirmed data up/down message** : End device does not need to acknowledge
 - **Confirmed data up/down message** : A message that must be acknowledged
 - **Uplink messages**: These messages are sent from endpoints to the network server and are relayed by one or more LoRaWAN gateways
 - **Downlink messages**: These messages flow from the network server to a single endpoint and are relayed by only a single gateway.
- LoRaWAN endpoints are uniquely addressable through a variety of methods.
- 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.
- 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.

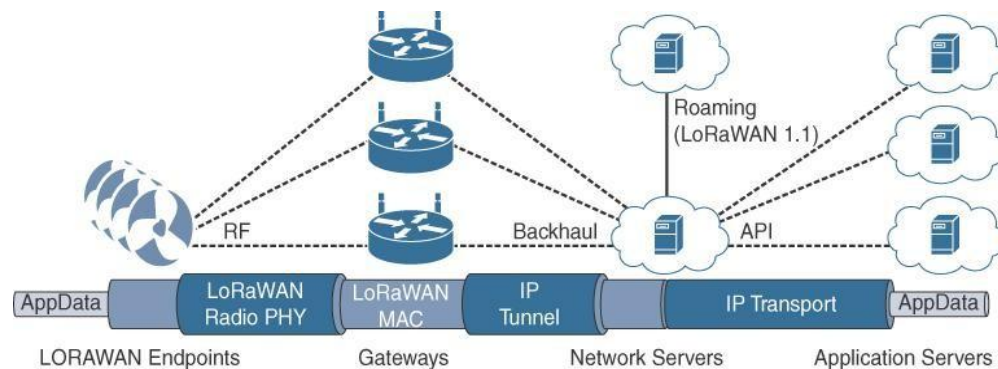


Figure 2.18: LoRaWAN Architecture

- In the figure 2.18 LoRaWAN endpoints transport their selected application data over the LoRaWAN MAC layer on top of one of the supported PHY layer frequency bands.
- 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.

❖ Security:

- LoRaWAN endpoints must implement two layers of security, protecting communications and data privacy across the network.
- Security in a LoRaWAN deployment applies to different components of the architecture as shown in figure 2.19

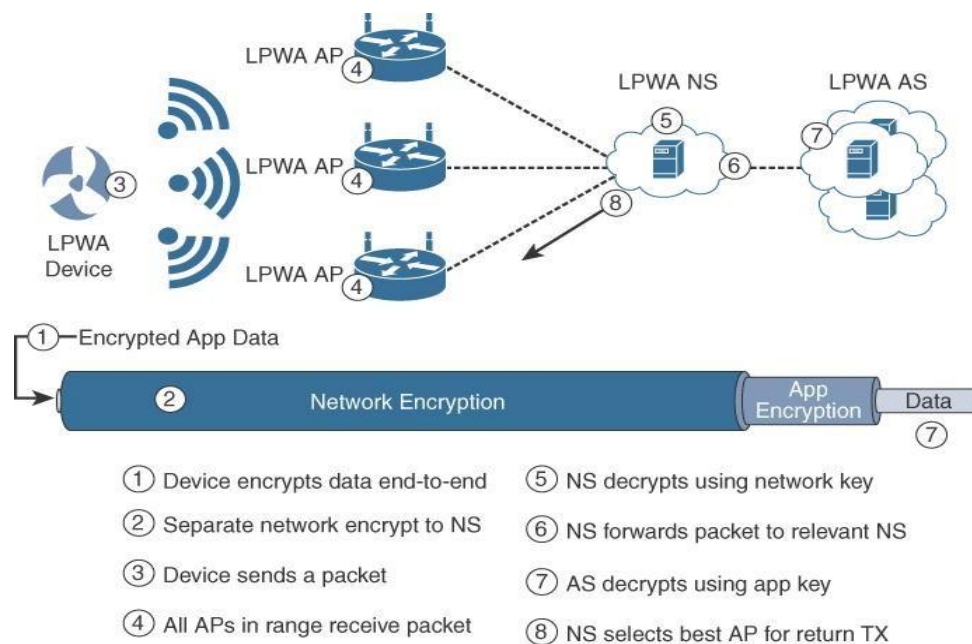


Figure 2.19: LoRaWAN Security

- 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.
- 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.

NB-IoT and Other LTE Variations:

- 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.

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.
- Category 0 includes important characteristics to be supported by both the network and end devices. 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.
- 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.

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.
 - **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.

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
- 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.
 - **Guard band:** An NB-IoT carrier is between the LTE or WCDMA bands. This requires coexistence between LTE and NB-IoT bands.

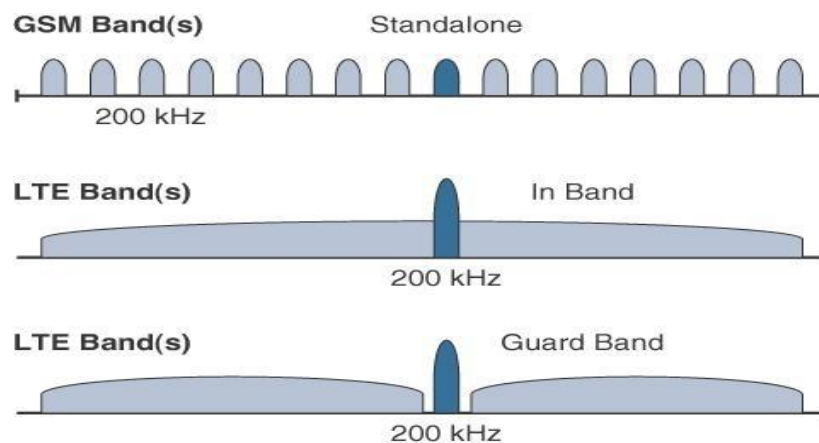


Figure 2.20: NB-IoT Deployment Options

- 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.
- 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.

Main Characteristics of Access Technologies is given in Table 2.5

Characteristic	IEEE 802.15.4g and					
	IEEE 802.15.4	IEEE 802.15.4e	IEEE 1901.2a	IEEE 802.11ah	LoRaWAN	NB-IoT
Wired or wireless	Wireless	Wireless	Wired	Wireless	Wireless	Wireless
Frequency bands	Unlicensed 2.4 GHz and sub-GHz	Unlicensed 2.4 GHz and sub-GHz	Unlicensed CENELEC A and B, FCC, ARIB	Unlicensed sub-GHz	Unlicensed sub-GHz	Licensed
Topology	Star, mesh	Star, mesh	Mesh	Star	Star	Star
Range	Medium	Medium	Medium	Medium	Long	Long
Data rate	Low	Low	Low	Low-high	Low	Low

Table 2.5 : Characteristics of Access Technologies

Module-3

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 ex-ample, fog computing).
- The system 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.

- **The Key Advantages of Internet Protocol**
 - 1) **Open and standards-based:** Operational technologies have often been delivered as turn key features by vendors who may have optimized the communication through closed and proprietary networking solutions.
 - 2) **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 IoT data through backhaul links and in the data center.
 - 3) **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.
 - 4) **Scalable:** As the common protocol of the Internet, IP has been massively deployed and tested for robust scalability.
 - 5) **Manageable and highly secure:** Communications infrastructure requires appropriate management and security capabilities for proper operations. Well-known network and security management tools are easily leveraged with an IP network layer.
 - 6) **Stable and resilient:** IP has a large and well-established know-ledge base and, more importantly, it has been used for years in critical infrastructures, such as financial and defense networks.
 - 7) **Consumers’ market adoption:** When developing IoT solutions and products targeting the consumer market, vendors know that consumers access to applications and devices will occur predominantly over broad-band and mobile wireless infrastructure.

- 8) **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. IP is a standard based protocol that is ubiquitous, scalable, versatile, and stable

- **Adoption or Adaptation of the Internet Protocol**

- The use of numerous network layer protocols in addition to IP is often a point of contention between computer networking experts.
- Adaptation means application layered gateways must be implemented to ensure the translation between non-IP and IP layer
- *Adoption* involves replacing all non-IP layers with their IP layer counter parts, simplifying the deployment model and operations.
- 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.
- We should consider the following factors when trying to determine which model is best suited for last-mile connectivity:
 - 1) **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
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 more difficult.
 - 2) **Overhead for last-mile communications paths:** IP adoption implies a layered architecture with a per-packet overhead that varies depending on the IP version. 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.
 - 3) **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 may put controls and limits on the “end-to-end” concept.

- 4) **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. 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.

THE NEED FOR OPTIMIZATION

The Internet of Things will largely be built on the Internet Protocol suite. However, challenges still exist for IP in IoT solutions. In addition to coping with the integration of non-IP devices, we may need to deal with the limits at the device and network levels that IoT often imposes. Therefore, optimizations are needed at various layers of the IP stack to handle the restrictions that are present in IoT networks.

- **Constrained Nodes**

Depending on its functions in a network a “thing” architecture may or may not offer similar characteristics compared to a generic PC or server in an IT environment.

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, you may implement 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).
- ❖ **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.

- **Constrained Networks**

- ❖ Network bandwidth capacity was restrained due to technical limitations. Connections often depended on low-speed modems for transferring data.
- ❖ 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 utilize a star, mesh, or combined network topologies, ensuring proper operations.
- ❖ Control plane traffic must also be kept at a minimum; otherwise, it consumes the bandwidth that is needed by the data traffic.
- ❖ The power consumption in battery-powered nodes must be considered. Any failure or verbose control plane protocol may reduce the lifetime of the batteries.
- ❖ Constrained nodes and networks pose major challenges for IoT connectivity in the last mile. This in turn has led various standards organizations to work on optimizing protocols for IoT.

- **IP Versions**

- ❖ The IETF (The Internet Engineering Task Force) 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.

. 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.
- ❖ **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
- ❖ **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
- ❖ **IPv6 Adaptation Layer:** IPv6-only adaptation layers for some physical and data link layers for recently standardized IoT protocols support only IPv6. While the most common physical and data link layers (Ether-net, Wi-Fi, and so on) specify 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).

OPTIMIZING IP FOR IOT

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

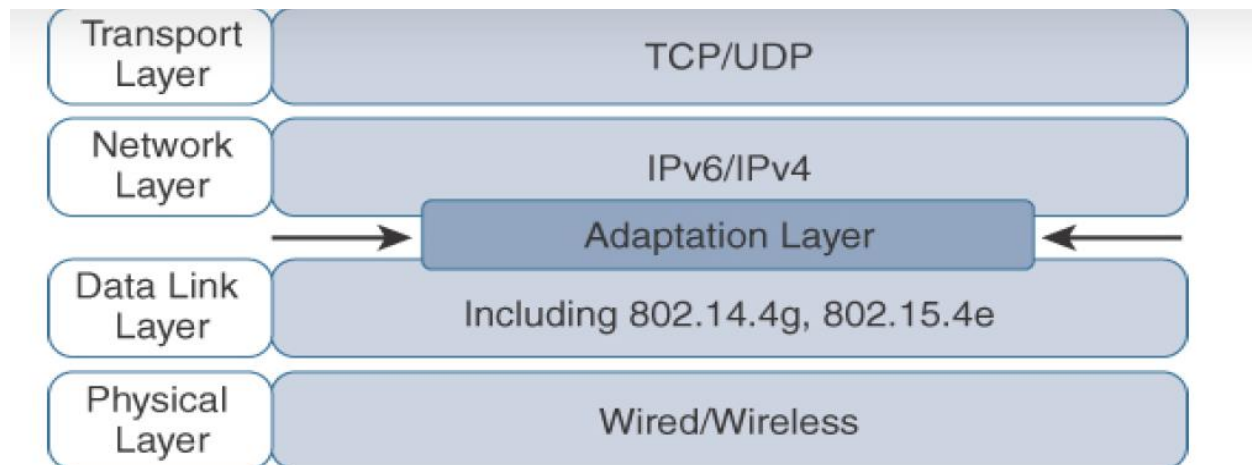


Figure: Optimizing IP for IOT using an Adaptation Layer

- **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*.
- ❖ The main examples of adaptation layers optimized for constrained nodes or “things” are the ones under the 6LoWPAN working 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.

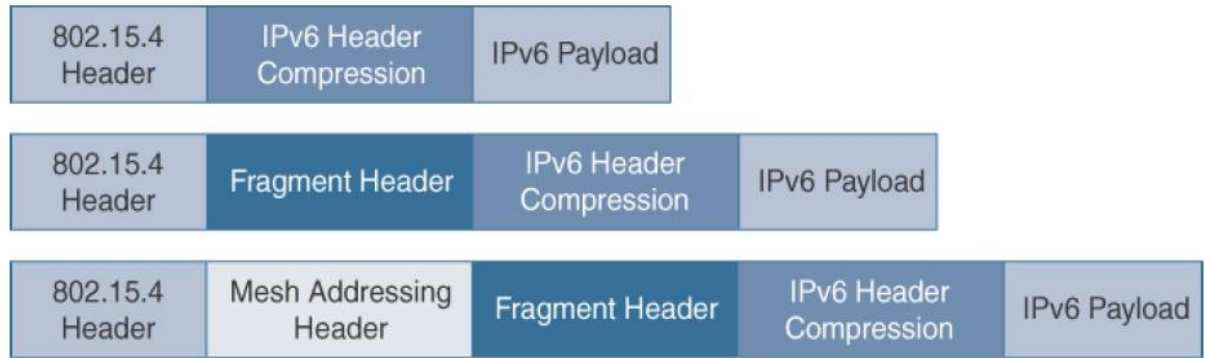


Figure: 6LoWPAN Header Stack

The above figure shows the sub-headers related to compression, fragmentation and mesh addressing.

- **Header Compression**

- ❖ IPv6 header compression for 6LoWPAN was defined initially in RFC 4944 and subsequently updated by RFC 6282
- ❖ Note that header compression for 6LoWPAN is only defined for an IPv6 header and not IPv4
- ❖ 6LoWPAN header compression is stateless, and conceptually it is not too complicated.
- ❖ 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 cases

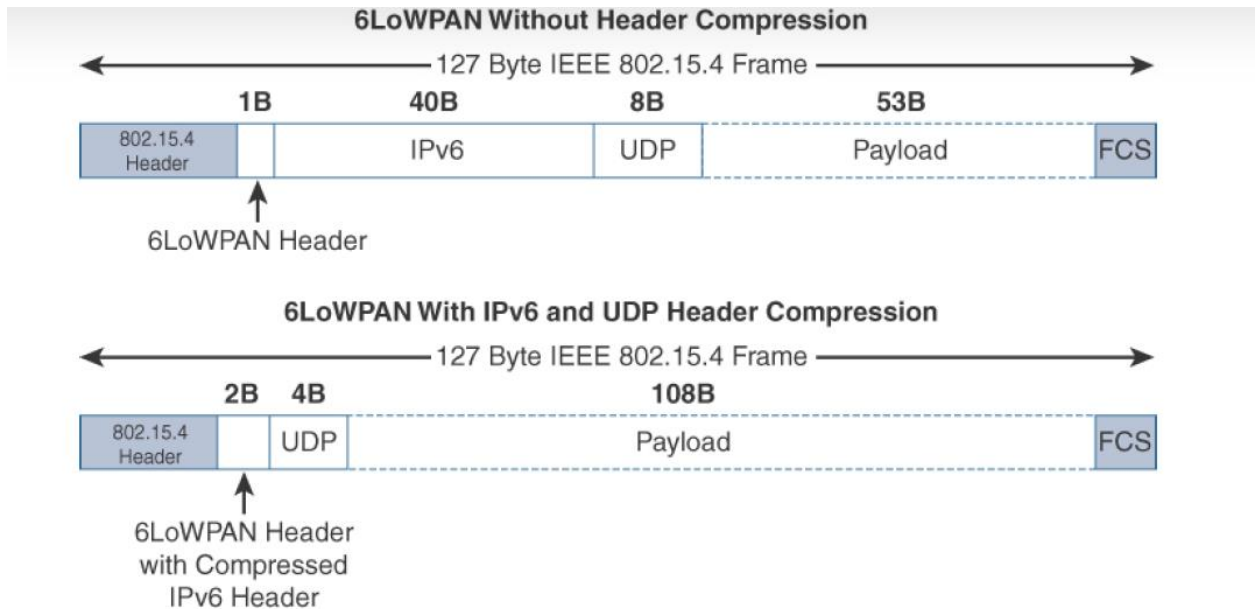


Figure: 6LoWPAN Header Compression

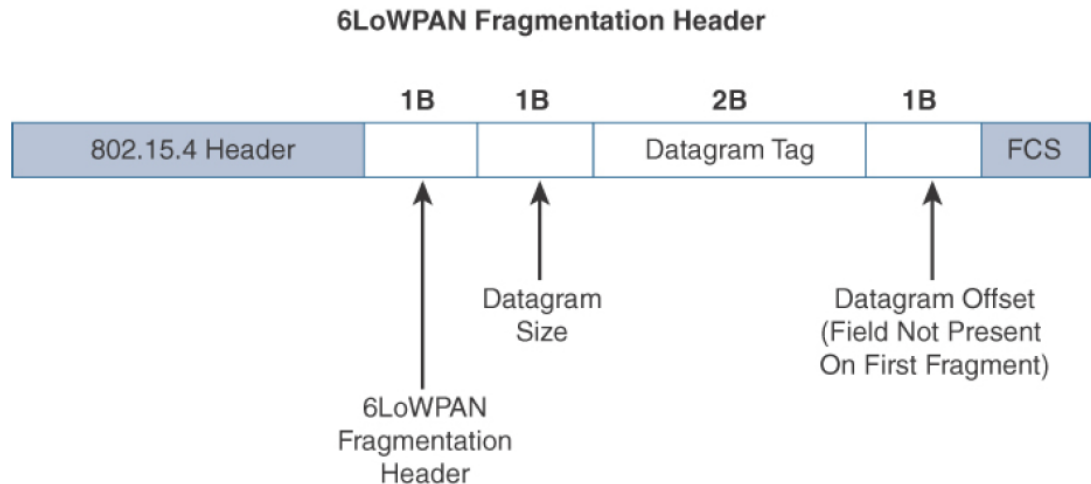
- ❖ At the top of Figure we can see a 6LoWPAN frame without any header compression enabled: The full 40-byte IPv6 header and 8-byte UDP header are visible. The 6LoWPAN header is only a single byte in this case. Notice that uncompressed IPv6 and UDP headers leave only 53 bytes of data payload out of the 127-byte maximum frame size in the case of IEEE802.15.4.
- ❖ The bottom half of Figure shows a frame where header compression has been enabled for a best-case scenario. UDP has been reduced in half, to 4 bytes from 8. The compressed 40 byte IPv6 header is put in the 2bytes header section. Most importantly, the header compression has allowed the payload to more than double, from 53 bytes to 108 bytes, which is obviously much more efficient.

• 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
- ❖ The fragment header utilized by 6LoWPAN is composed of three primary fields: Datagram Size (total size of un-fragmented payload), Datagram Tag (identifies set

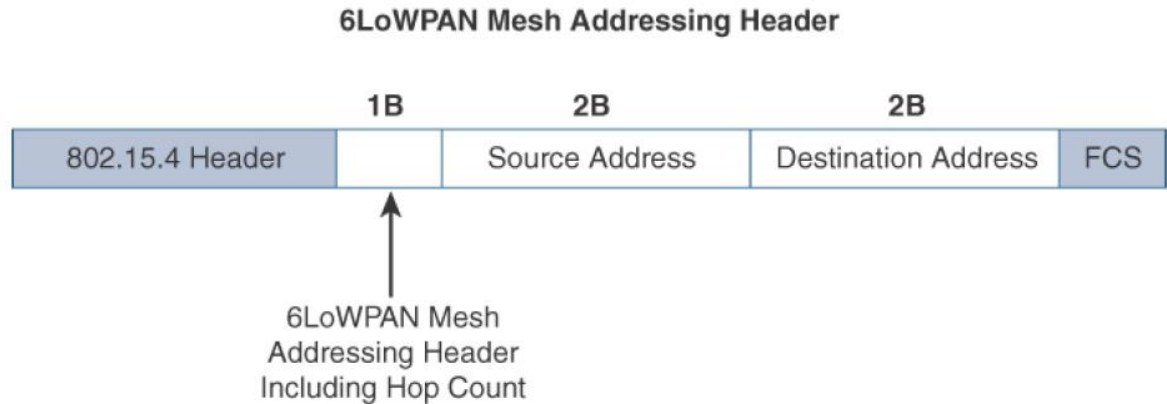
of fragments), and Datagram Offset(indicates how many fragments are remaining).

- ❖ The 6LoWPAN fragmentation header field itself uses a unique bit value to identify that the subsequent fields behind it are fragment fields
- ❖ The first fragmentation header for an IPv6 payload being only 4 bytes long. The remainder of the fragments have a 5-byte header field



✓ Mesh Addressing

- ❖ The purpose of the 6LoWPAN mesh addressing function is to forward packets over multiple hops.
- ❖ Three fields are defined for this header: Hop Limit, Source Address, and Destination Address.
- ❖ 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.



❖ 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.

✓ **6TiSCH**

- ❖ 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 Wireless HART and ISA100.11a.
- ❖ Devices implementing IEEE 802.15.4e TSCH communicate by following a Time Division Multiple Access (TDMA) schedule.

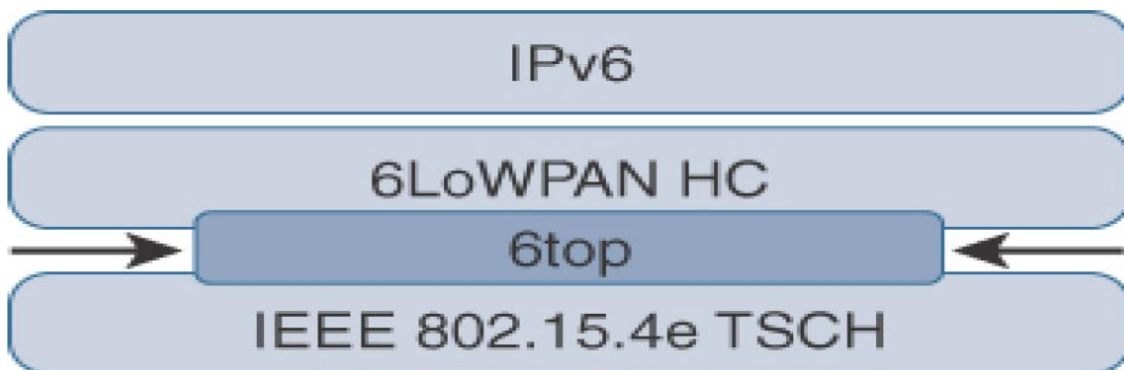


Figure: Location of 6TiSCH'S 6 TOP Sublayer

- ❖ 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 management mechanisms
 - **Static scheduling:** All nodes in the constrained 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 may expect 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 provides quite a bit of flexibility and control in allocating cells for communication between nodes.
 - **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 the path.
- ❖ There are three 6TiSCH forwarding models:
 - **Track Forwarding (TF):** This is the simplest and fastest forwarding model. A “track” in this model is 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.

- **Fragment forwarding (FF):** This model takes advantage of 6LoWPAN fragmentation to build a Layer 2 forwarding table. The 6LoWPAN sublayer 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.

✓ **RPL**

- ❖ The new distance-vector routing protocol was named the IPv6 Routing Protocol for Low Power and Lossy Networks (RPL).

The RPL specification was published as RFC 6550 by the RoLL.

- ❖ 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
- ❖ 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 do-main. Every node knows how to directly reach every other node.
 - **Non-storing mode:** Only the border router(s) of the RPL domain contain 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.

RPL is based on the concept of a directed acyclic graph (DAG). A DAG is a directed graph where no cycles exist.

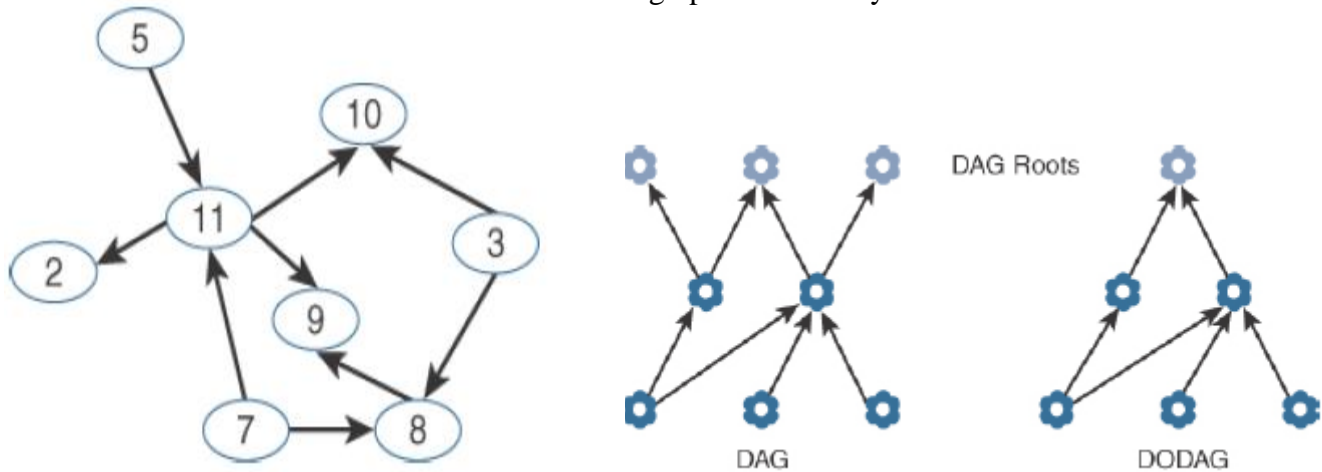


Figure: Direct Acyclic Graph and DODAG

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 compares a DAG and a DODAG. We can see that that a DAG has multiple roots, whereas the DODAG has just one.

In a DODAG, each node maintains up to three parents that provide a path to the root. Typically, one of these parents is the preferred parent, which means it is the preferred next hop for upward routes toward the root.

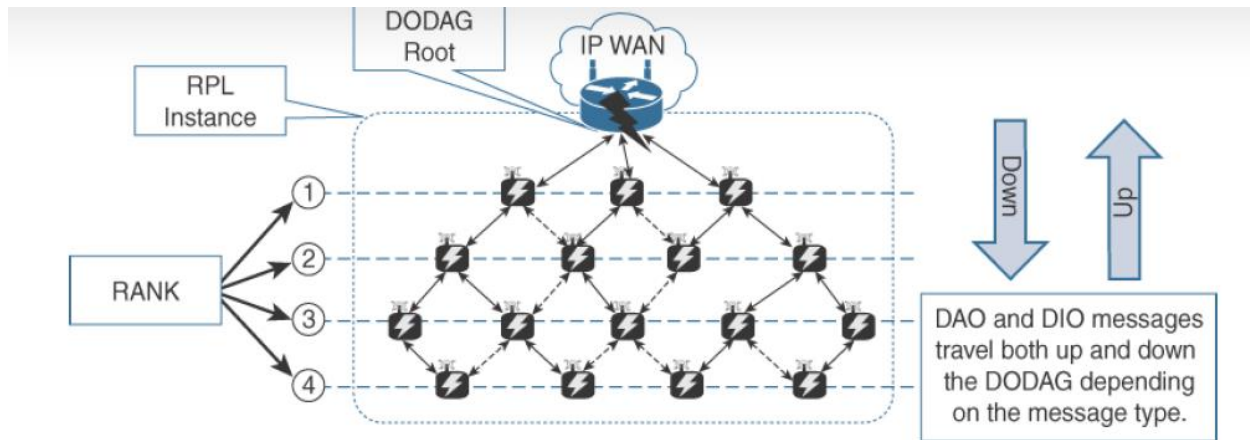


Figure: RPL overview

- Upward routes in RPL are discovered and configured using DAG Information Object (DIO) messages.
- The information in DIO messages determines parents and the best path to the DODAG root.
- Destination Advertisement Object (DAO) message. DAO messages allow nodes to inform their parents of their presence and reach ability to descendants.
- An objective function (OF) defines how metrics are used to select routes and establish a node's 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.
- Specific network layer headers are defined for datagrams being forwarded within an RPL domain.
- RFC 6554 specifies the Source Routing Header (SRH) for use between RPL routers. A border router or DODAG root inserts the SRH when specifying a source route to deliver datagrams to nodes downstream in the mesh network
- RPL defines a large and flexible set of new metrics and constraints for routing in RFC 6551. Some of the RPL routing metrics and constraints defined in RFC6551 include the following:

- ✓ **Expected Transmission Count (ETX):** Assigns a discrete value to the number of transmissions a node expects to make to deliver a packet.
- ✓ **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 by high 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. Of-ten, nodes conserving power use lower throughput. This metric allows the prioritization of paths with higher throughput.

Authentication and Encryption on Constrained Nodes

IETF has mentioned two security working groups: ACE and DICE.

❖ ACE

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.

DICE

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.

PROFILES AND COMPLIANCES

The Internet Protocol suite for smart objects involves a collection of protocols and options that must work in coordination with lower and upper layers. Therefore, profile definitions, certifications, and promotion by alliances can help implementers develop solutions that guarantee interoperability and/or inter-changeability of devices.

Some of the main industry organizations working on profile definitions and certifications for IoT constrained nodes and networks.

❖ **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. 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.

❖ **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.

❖ **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

❖ **IPv6 Ready Logo**

The IPv6 Ready Logo program has established conformance and inter operability 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.

CHAPTER2: APPLICATION PROTOCOLS FOR IOT

THE TRANSPORT LAYER

IoT networks which supports TCP/IP architecture uses two main protocols in transport layer.

Transmission Control Protocol (TCP): This connection-oriented protocol requires a session to get established between the source and destination before exchanging data.

User Datagram Protocol (UDP): With this connectionless protocol, data can be quickly sent between source and destination—but with no guarantee of delivery.

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

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 must be guaranteed error free, the application layer protocol takes care of that function.

Use of TCP in constrained IOT platform and high data rate environments is highly challenging.

IOT APPLICATION TRANSPORT METHODS

The different types of IoT application protocols have various means for transporting these protocols across a network. The following categories of IoT application protocols and their transport methods are considered.

- ✓ Application layer protocol not present
- ✓ Supervisory control and data acquisition (SCADA)
- ✓ Generic web-based protocols
- ✓ IoT application layer protocols

❖ Application layer protocol not present

IETF RFC 7228 devices defined as class 0 send or receive only a few bytes of data. Class 0 devices are usually simple smart objects that are severely constrained.

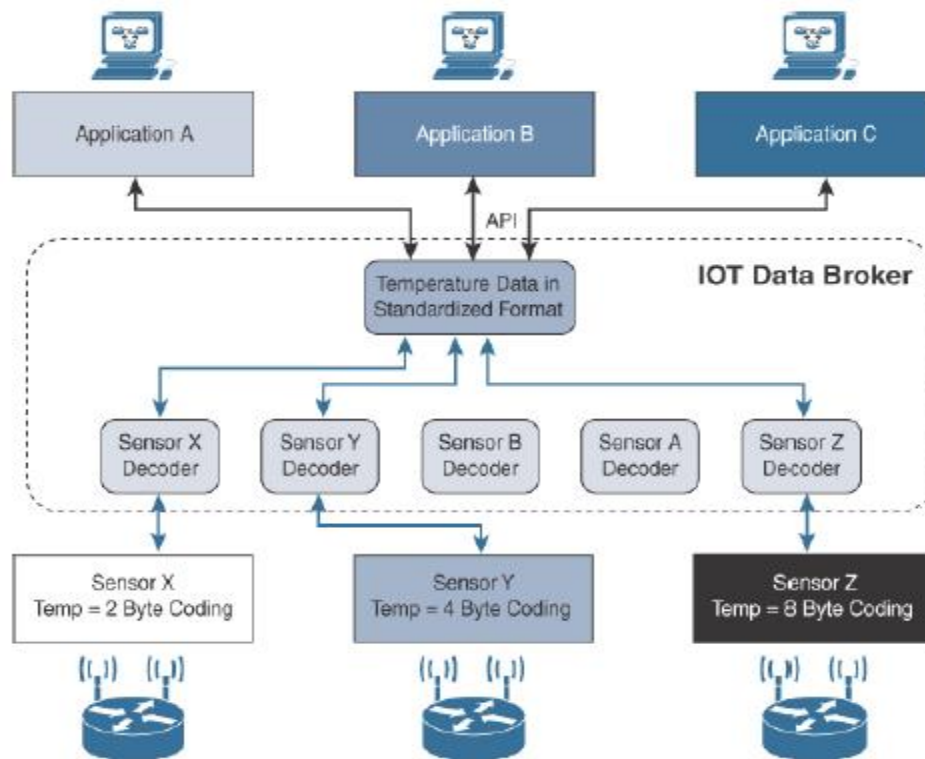


Figure: IOT Broker

In the figure, the different kinds of temperature sensors from different manufacturers are used. These sensors will report temperature data in varying formats. If we increase the number of sensors upto thousands in the application. The interpreting the received temperature in different formats becomes complex. The solution to this problem is to use IOT Broker.

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.

In the figure, 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 can access this temperature data without having to deal with decoding multiple temperature data formats.

❖ **Supervisory control and data acquisition (SCADA)**

Combined with the fact that IP is the default standard for computer networking in general, older protocols that connected sensors and actuators have evolved and adapted them to utilize IP.

At a high level, SCADA systems collect sensor data and telemetry from re-mote 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

• **Adapting SCADA for IP**

In the 1990s, the rapid adoption of Ethernet networks in the industrial world led to the evolution of SCADA application layer protocols.

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.

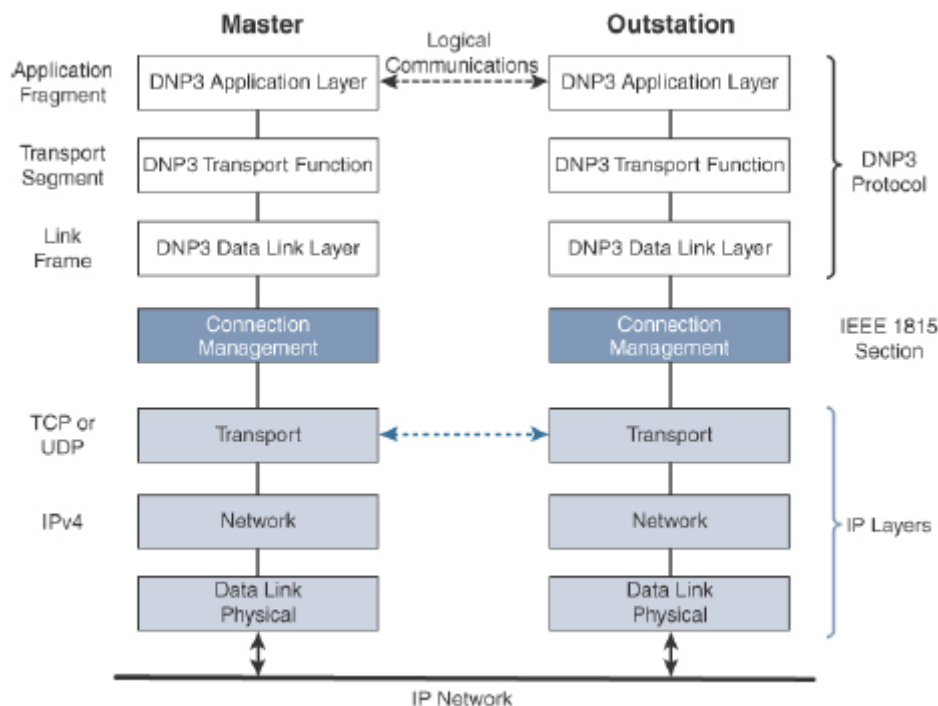


Figure: Protocol Stack for Transporting Serial DNP3(Distributed Network protocol)

- 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.
- 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 DNP3 end points or devices are not aware of the underlying IP transport that is occurring.
- 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 processes that can both listen for connection requests and perform an active open on the channel if required.
- 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.

- **SCADA Protocol Translation**

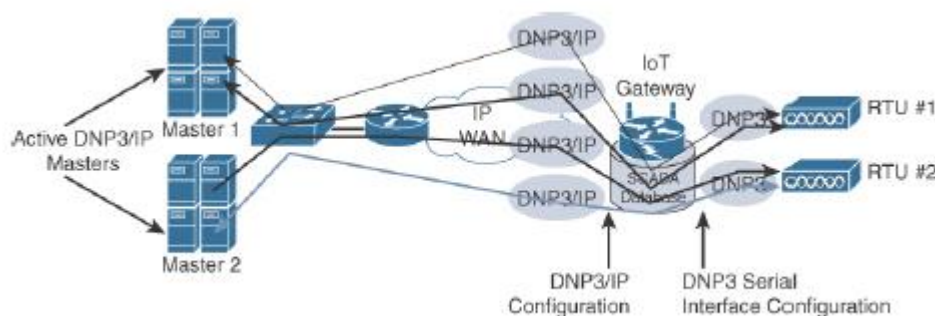
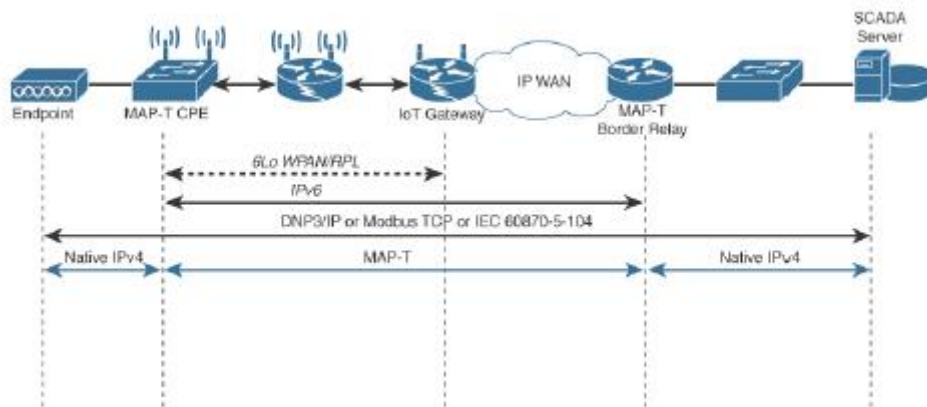


Figure: DNP3 translation

- Figure 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.
- By running protocol translation, the IoT gateway connected to the RTUs in Figure 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.

- **SCADA Transport over LLNs with MAP-T**

**Figure: DNP3 Protocol over 6LOWPAN Networks with MAP-T**

- The above figure shows a scenario in which a legacy endpoint is connected across an LLN (Low Power and Lossy Network) running 6LoWPAN (IPv6 over Low Power Wireless Personal Area Network) 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 case the legacy devices and the SCADA server support only IPv4 and IPv6 is being used for connectivity to the endpoint.
- 6LoWPAN is 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 only IPv6.
- The solution to this problem is to use the protocol known as MAP-T
- 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.

❖ Generic Web-Based Protocols

- Web based protocols have become common in consumer and enterprise applications and services. Therefore, it makes sense to try to use these protocols when developing IoT applications, services, and devices in order to ease the integration of data and devices from prototyping to production.
- 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
- 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).

❖ IoT Application Layer Protocols

- When considering constrained networks and/or a large-scale deployment of constrained nodes, verbose web-based and data model protocols, may be too heavy for IoT applications.
- Two of the most popular protocols are MQTT and CoAP.

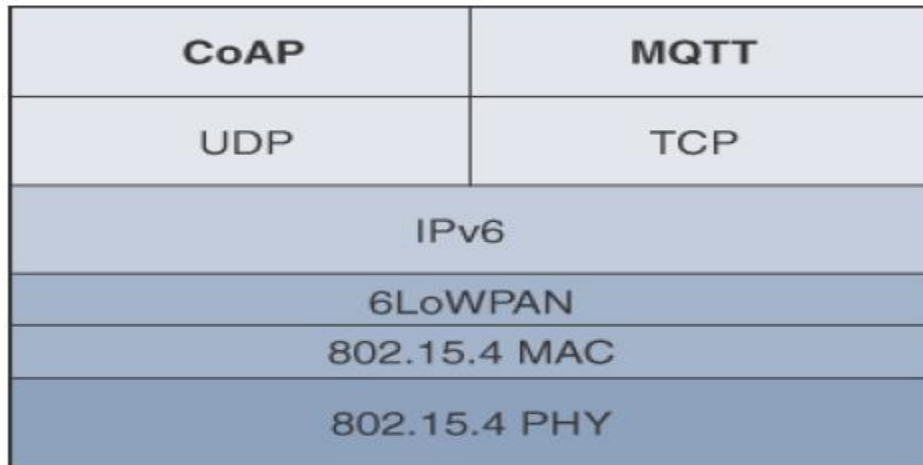


Figure: Example of a High-Level IoT Protocol Stack for CoAP and MQTT

In the above figure, CoAP and MQTT are naturally at the top of this sample IoT stack, based on an IEEE 802.15.4 mesh network.

✓ 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).

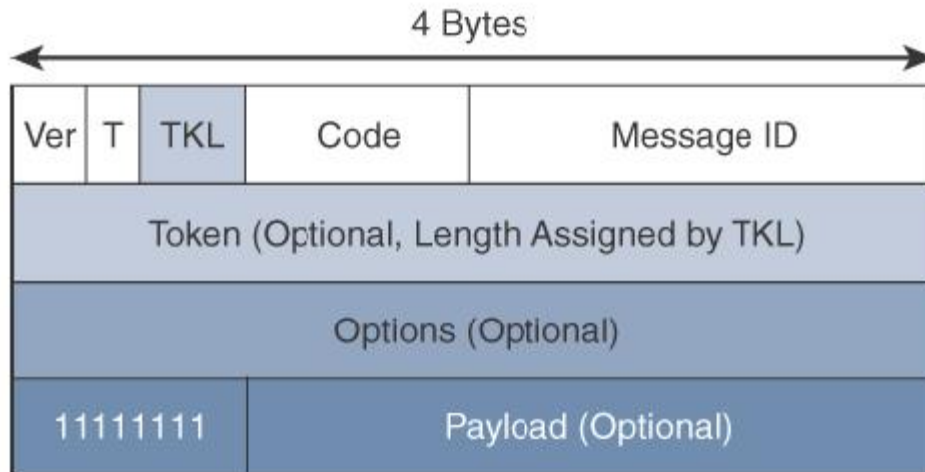


Figure: CoAP Message Format

In the above figure, 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 con-strained devices.

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.

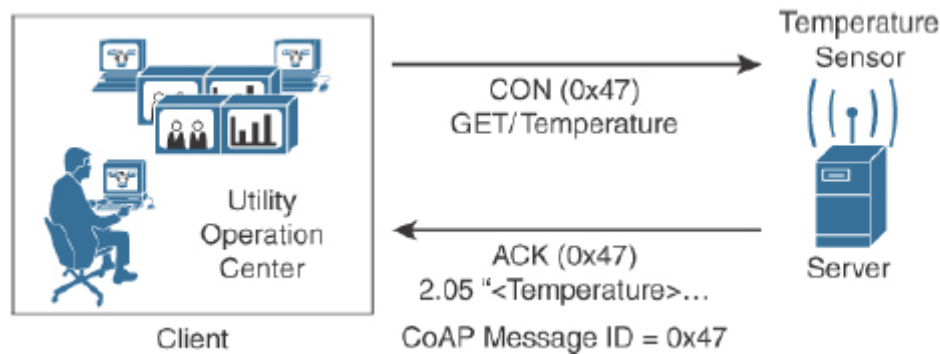
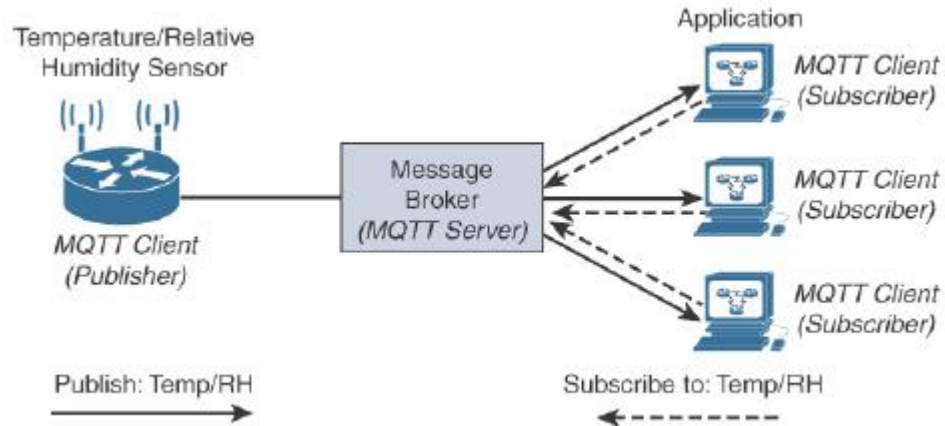
Table: CoAP Message Fields**Figure: CoAP Reliable Transmission Example**

Figure 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.

✓ Message Queuing Telemetry Transport (MQTT)

- At the end of the 1990s, engineers from IBM and Arcom (acquired in 2006 by Eurotech) 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.
- They have introduced a client/server and publish/ subscriber framework based on the TCP/IP architecture.

- An MQTT client can act as a publisher to send data (or resource information) to an MQTT server acting as an MQTT message broker.



- The above figure shows the MQTT Publish/Subscribe framework
- The MQTT client on the left side of figure is a temperature (Temp) and relative humidity (RH) sensor that publishes its Temp/RH data.
- The MQTT server (or message broker) accepts the net-work 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 is an MQTT client that is a subscriber to the Temp/RH data being generated by the publisher or sensor on the left.
- 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.
- MQTT is a lightweight protocol because each control packet consists of a 2-byte fixed header with optional variable header fields and optional pay-load.

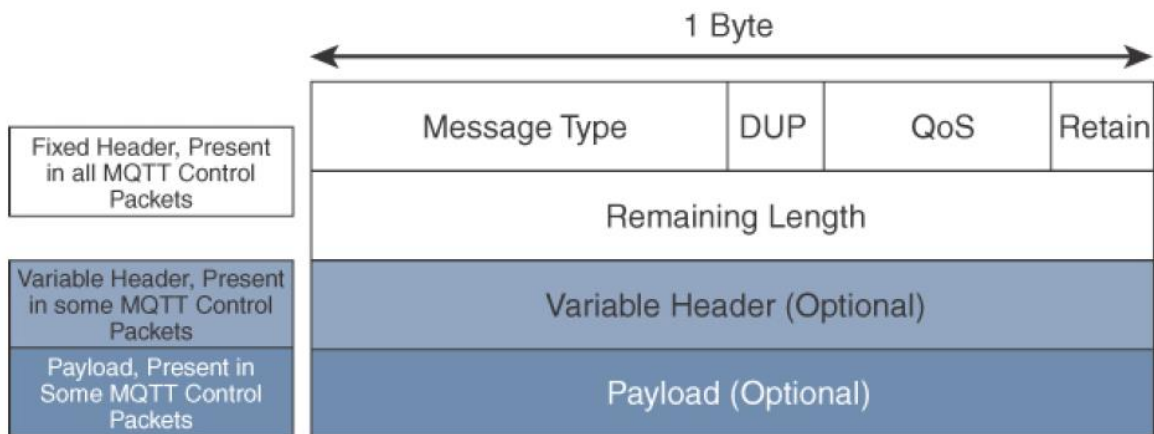


Figure: MQTT message format

- Compared to the CoAP message format in, 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 the below Table

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

Table: MQTT Message Types

- 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. These are discussed in more detail later in this chapter.
- 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 this field.
- MQTT sessions between each client and server consist of four phases: session establishment, authentication, data exchange, and session termination. Each client connecting to a server has a unique client ID.
- The MQTT protocol offers three levels of quality of service (QoS)
- ✓ **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 the variable header. If the message is not acknowledged by a PUBACK packet, 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. The packet contains an optional variable header with a packet identifier. 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.

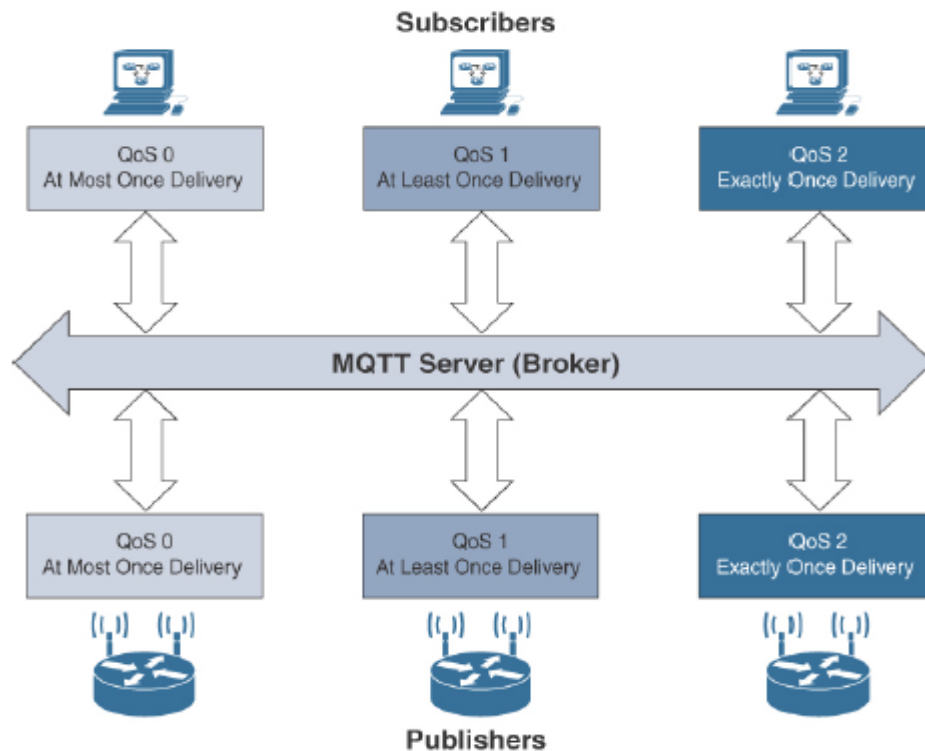


Figure: MQTT QoS Flow

❖ Differences 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

CHAPTER-1

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.

Before diving deeper into data analytics, it is important to define a few key concepts related to data. Depending on how data is categorized, various data analytics tools and processing methods can be applied.

Structured Versus Unstructured Data

Structured data and unstructured data are important classifications as they typically require different toolsets from a data analytics perspective.

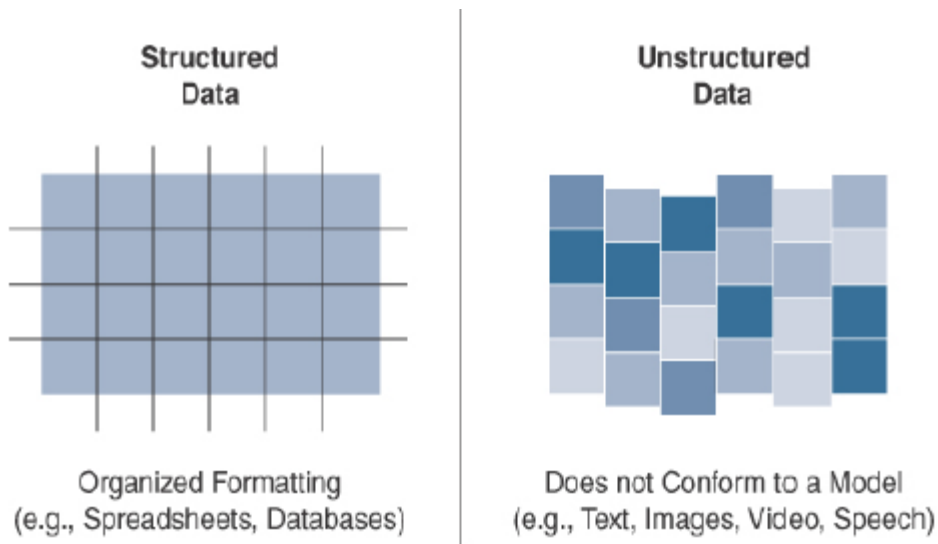


Figure: Comparison between structured and unstructured Data

- 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).
- Structured data can be found in most computing systems and includes everything from banking transaction and invoices to computer log files and router configurations.
- Unstructured data lacks a logical schema for understanding and decoding the data through traditional programming means.

Data in Motion Versus Data at Rest

Data in IoT networks is either in transit (“data in motion”) or being held or stored (“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 arrives at the data center, it is possible to process it in real-time, just like at the edge, while it is still in motion.

Data at rest in IoT networks can be typically found in IoT brokers or in some sort of storage array at the data center. The best known of these tools is Hadoop.

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 in-sights.

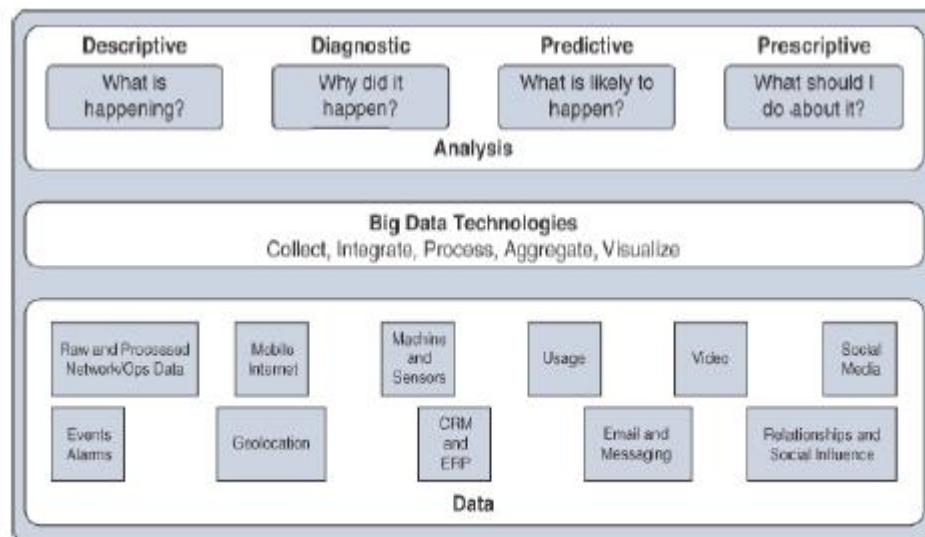


Figure: Types of data analysis results

- **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.
- **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.
- **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.

- **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.

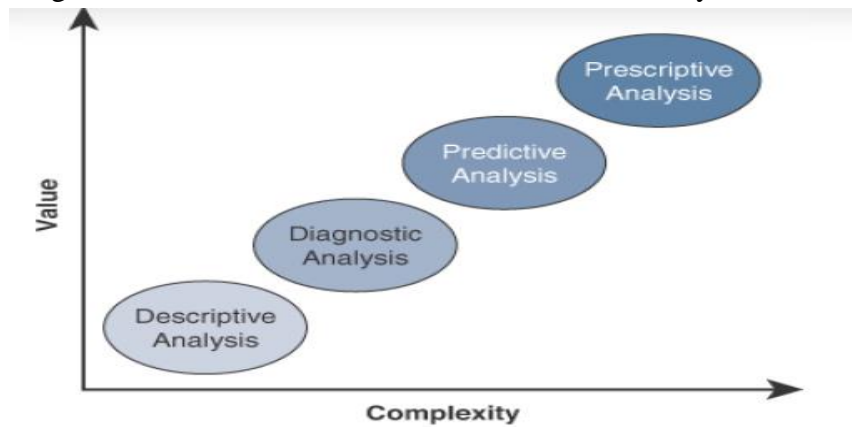


Figure: Application of value and complexity factors to the types of data analysis

IoT Data Analytics Challenges

As IoT has grown and evolved, it has become clear that traditional data analytics solutions were not always adequate.

IoT data places two specific challenges on a relational database

- **Scaling problems:** Due to the large number of smart objects in most IoT networks that continually send data, relational databases can grow in-credibly large very quickly.
- **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.

MACHINE LEARNING

Machine learning, deep learning, neural networks and convolutional networks are words you have probably heard in relation to big data and IoT. ML is indeed central to IoT.

Machine Learning Overview

Machine learning is, in fact, part of a larger set of technologies commonly grouped under the term *artificial intelligence (AI)*.

AI includes any technology that allows a computing system to mimic human intelligence using any technique, from very advanced logic to basic “if-then-else” decision loops.

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.

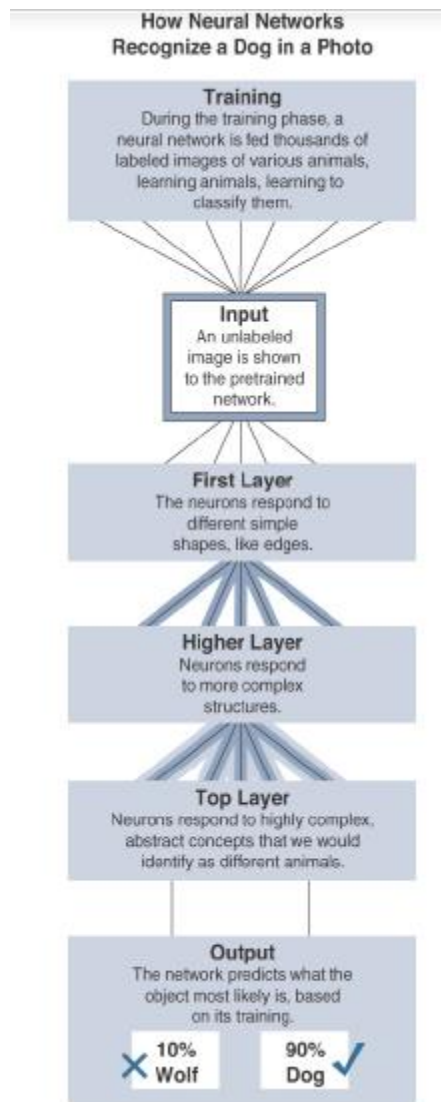
- With supervised learning techniques, hundreds or thousands of images are fed into the machine, and each image is labeled. This is called the *training set*.
- Each new image is compared to the set of known “good images,” and a deviation is calculated to determine how different the new image is from the average human image. 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 un-labeled 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.
- For example grouping of engines by the sound they make at a given temperature.
- 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 for manual 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.

Neural Networks

- Processing multiple dimensions requires a lot of computing power. It is also difficult to determine what parameters to input and what combined variations should raise red flags.
- Neural networks are ML methods that mimic the way the human brain works.
- 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.
- The great efficiency of neural networks is that each unit processes a simple test, and therefore computation is quite fast.



For ex-ample, 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

Machine Learning and Getting Intelligence from Big Data

ML operations into two broad subgroups

- **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).
- **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.

The common applications of ML for IoT revolve around four major domains

- ❖ **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 fail-ure conditions or to better evaluate the environment.
- ❖ **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.
- ❖ **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. 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 the operations.
- ❖ **Self-healing, self-optimizing:** A fast-developing aspect of deep learning is the closed loop. ML-based monitoring triggers changes in machine behavior and operations optimizations. The ML engine can be programmed to dynamically monitor and combine new parameters and automatically deduce and implement new optimizations when the results demonstrate a possible gain. The system becomes self-learning and self-optimizing.

BIG DATA ANALYTICS TOOLS AND TECHNOLOGY

Big data analytics can consist of many different software pieces that together collect, store, manipulate and analyze all different data types. Generally, the in-dustry looks to the “three Vs” 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.
- **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 three types.
- **Volume:** Volume refers to the scale of the data. Typically, this is measured from gigabytes on the very low end to petabytes or even exa-bytes of data on the other extreme. 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 is transactional 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.

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.

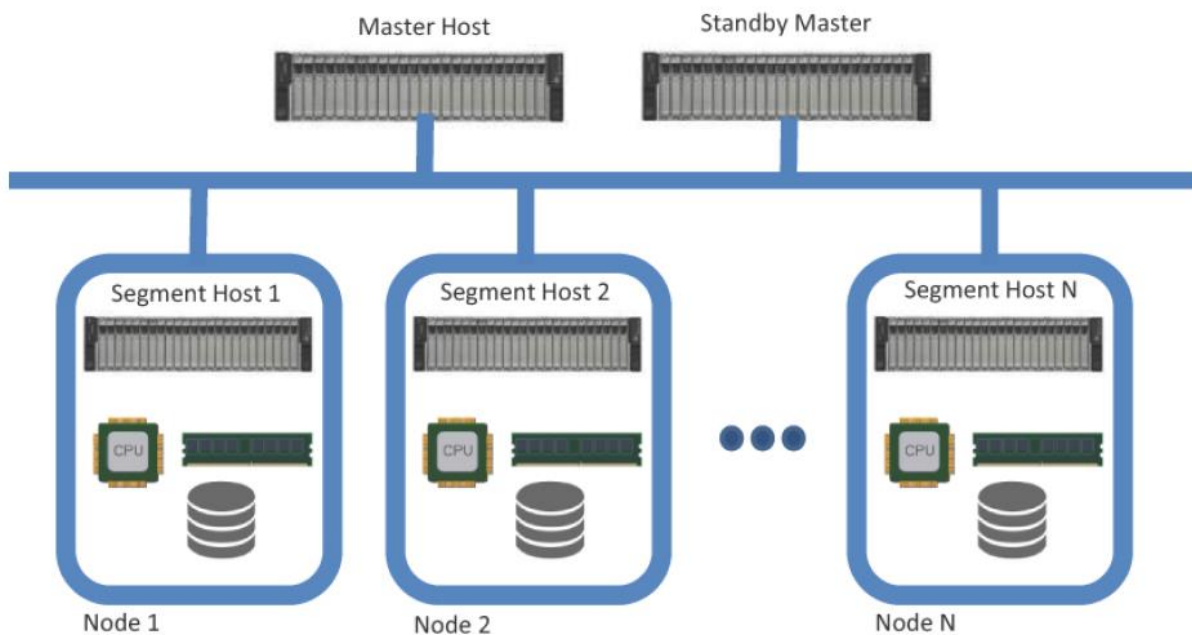


Figure: MPP shared nothing architecture

In the figure, we see it 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 “shared-nothing” fashion, with each node containing local processing, memory, and storage and operating in-dependently. Data storage is optimized across the nodes in a structured SQL-like format that allows data analysts to work with the data using common SQL tools and applications.

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

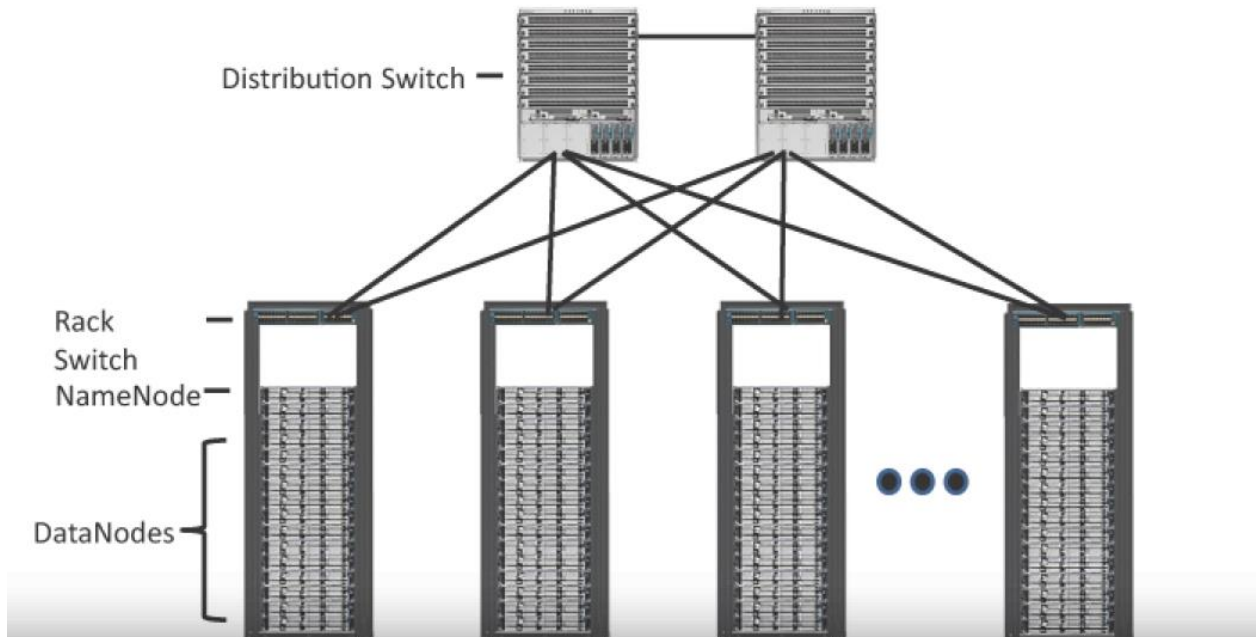
specific database technology; rather, it is an umbrella term that encompasses several different types of databases, including the following.

- **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 allow for 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.

HADOOP

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



For HDFS, this capability is handled by specialized nodes in the cluster, including NameNodes and DataNodes.

- **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 Name Node 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 NameNodes and DataNodes coordinate block-level redundancy with this replication technique.

EDGE STREAMING ANALYTICS

A major area of evolution for IT in the past few years has been the transition to cloud services

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.
- **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 CORE FUNCTIONS

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.

- **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.

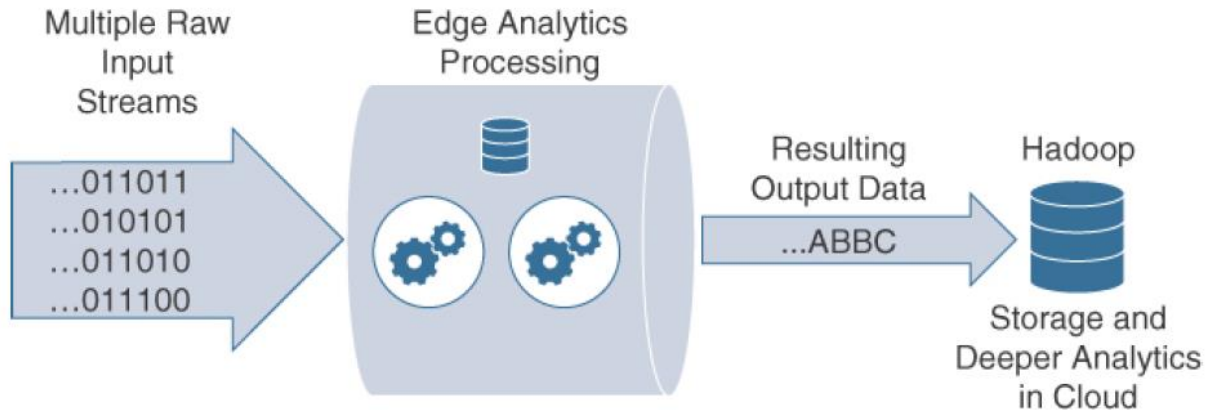


Figure: illustrates the stages of data processing in an edge APU

In order to perform analysis in real-time, the APU needs to perform the following functions

- **Filter:** The streaming data generated by IoT endpoints is likely to be very large, and most of it is irrelevant. For example, a sensor may simply poll on a regular basis to confirm that it is still reachable.
- **Transform:** In the data warehousing world, Extract, Transform, and Load (ETL) operations are used to manipulate the data structure into a form that can be used for other purposes.
- **Time:** As the real-time streaming data flows, a timing context needs to be established. This could be to correlated average temperature readings from sensors on a minute-by-minute basis.

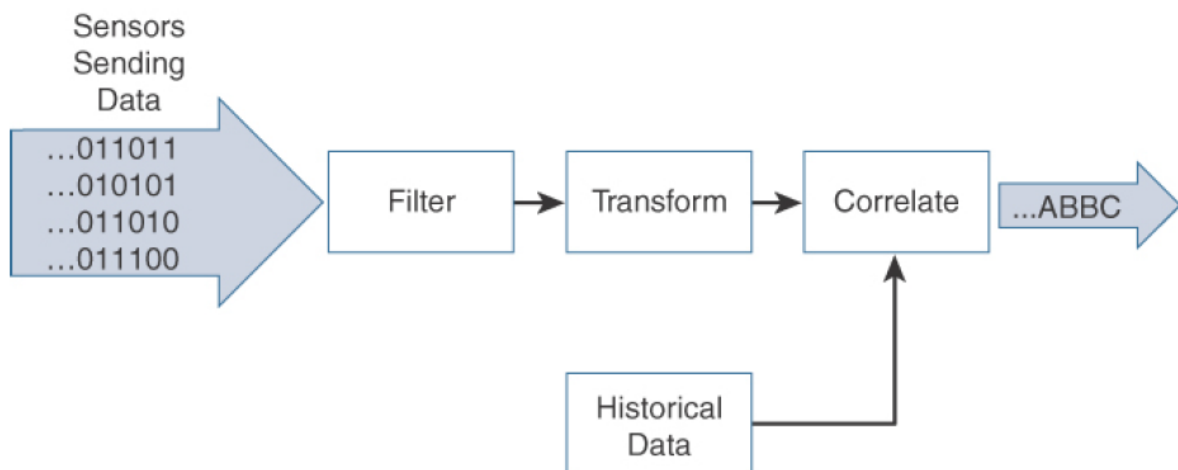


Figure: Corelating data stream with historical data

- **Correlate:** Streaming data analytics becomes most useful when multiple data streams are combined from different types of sensors. For example in hospital, several vital signs are measured for patients including body temperatures, blood pressure heart rate and respiratory rate and all the data are used to get the clear picture of patients health.
- **Match patterns:** Once the data streams are properly cleaned, trans-formed, and correlated with other live streams as well as historical datasets, pattern matching operations are used to gain deeper insights to the data. For example, If an unexpected event arises, such as a sudden change in heart rate or respiration, the pattern matching operator recognizes this as out of the ordinary and can take certain actions, such as generating an alarm to the nursing staff.
- **Improve business intelligence:** Ultimately, the value of edge analytics is in the improvements to business intelligence that were not previously available. For example conducting edge analytics on patients in a hospital allows staff to respond more quickly to the patient’s changing needs and also reduces the volume of data sent to the cloud.

DISTRIBUTED ANALYTICS SYSTEMS

Depending on the application and network architecture, analytics can happen at any point throughout the IoT system. Streaming analytics maybe performed directly at the edge, in the fog, or in the cloud data center.

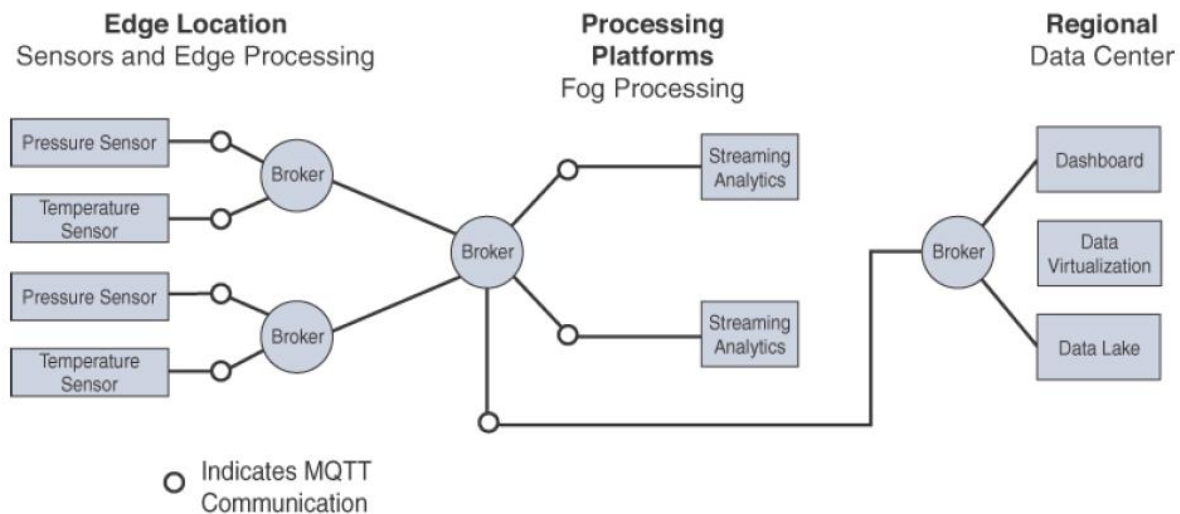


Figure: Distributed Analytics throughout the IoT system

The above figure 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. The fog node is located on the same oil rig and

performs streaming analytics from several edge devices, giving it better insights due to the expanded data set.

NETWORK ANALYTICS

Another form of analytics that is extremely important in managing IoT systems is network-based analytics. Network analytics has the power to analyze details of communications patterns made by protocols and correlate this across the network.

In addition to other network management services, are as follows:

- **Network traffic monitoring and profiling:** Flow collection from the network layer provides global and distributed near-real-time monitoring capabilities. IPv4 and IPv6 network wide traffic volume and pattern analysis helps administrators proactively detect problems and quickly troubleshoot and resolve problems when they occur.
- **Application traffic monitoring and profiling:** Monitoring and profiling can be used to gain a detailed time-based view of IoT access services, such as the application-layer protocols, including MQTT, CoAP, and DNP3, as well as the associated applications that are being used over the network.
- **Capacity planning:** Flow analytics can be used to track and anticipate IoT traffic growth and help in the planning of upgrades when deploying new locations or services by analyzing captured data over a long period of time.
- **Security analysis:** Because most IoT devices typically generate a low volume of traffic and always send their data to the same server(s), any change in network traffic behavior may indicate a cyber security event, such as a denial of service (DoS) attack.

FLEXIBLE NETFLOW ARCHITECTURE

Flexible NetFlow (FNF) and IETF IPFIX (RFC 5101, RFC 5102) are examples of protocols that are widely used for networks. FNF is a flow technology developed by Cisco Systems that is widely deployed all over the world. Key advantages of FNF are as follows:

- Flexibility, scalability, and aggregation of flow data
- Ability to monitor a wide range of packet information and produce new information about network behavior
- Enhanced network anomaly and security detection
- User-configurable flow information for performing customized traffic identification and ability to focus and monitor specific network behavior
- Convergence of multiple accounting technologies into one accounting mechanism

FNF Components

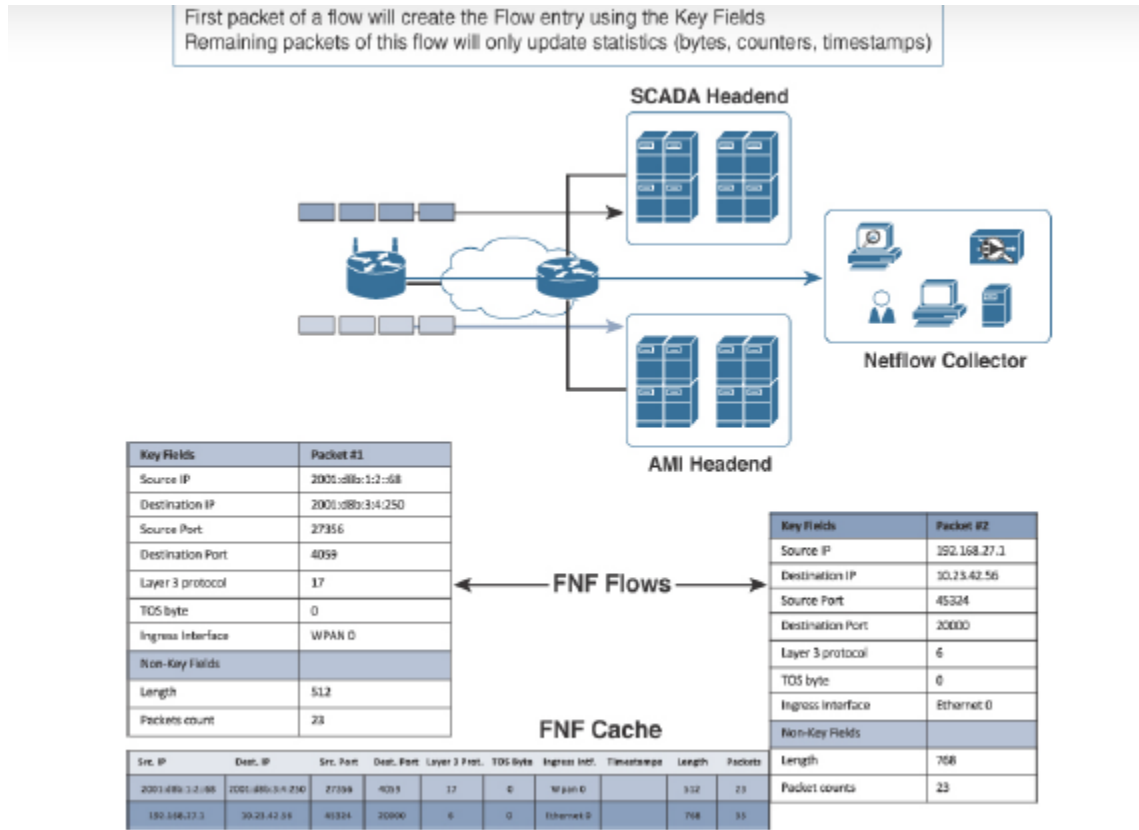


Figure:Flexible netflow overview

- FNF Flow Monitor (NetFlow cache):** The FNF Flow Monitor describes the NetFlow cache or information stored in the cache. The Flow Monitor contains the flow record definitions with key fields (used to create a flow, unique per flow record: match statement) and non-key fields (collected with the flow as attributes or characteristics of a flow) within the cache. Also, part of the Flow Monitor is the Flow Exporter, which contains information about the export of NetFlow information, including the destination address of the NetFlow collector. The Flow Monitor includes various cache characteristics, including timers for exporting, the size of the cache, and, if required, the packet sampling rate.
- FNF flow record:** A flow record is a set of key and non-key NetFlow field values used to characterize flows in the NetFlow cache. Flow records may be predefined for ease of use or customized and user defined. A typical predefined record aggregates flow data and allows users to target common applications for NetFlow. User-defined records allow selections of specific key or non-key fields in the flow record. The user-defined field is the key to Flexible NetFlow, allowing a wide range of information to be characterized and exported by NetFlow. It is expected that different network management applications

will support specific user-defined and predefined flow records based on what they are monitoring (for example, security detection, traffic analysis, capacity planning).

- **FNF Exporter:** There are two primary methods for accessing Net-Flow data: Using the **show** commands at the command-line interface (CLI), and using an application reporting tool. The Flexible NetFlow Exporter allows the user to define where the export can be sent, the type of transport for the export, and properties for the export. Multiple exporters can be configured per Flow Monitor.
- **Flow export timers:** Timers indicate how often flows should be exported to the collection and reporting server.
- **NetFlow export format:** This simply indicates the type of flow reporting format.
- **NetFlow server for collection and reporting:** This is the destination of the flow export. It is often done with an analytics tool that looks for anomalies in the traffic patterns.

Flexible NetFlow in Multiservice IoT Networks

In the context of multiservice IoT networks, it is recommended that FNF be configured on the routers that aggregate connections from the last mile's routers. Flow analysis at the gateway is not possible with all IoT systems. Some of the challenges with deploying flow analytics tools in an IoT network include the following

- The distributed nature of fog and edge computing may mean that traffic flows are processed in places that might not support flow analytics, and visibility is thus lost.
- IPv4 and IPv6 native interfaces sometimes need to inspect inside VPN tunnels, which may impact the router's performance.
- Additional network management traffic is generated by FNF reporting devices. The added cost of increasing bandwidth thus needs to be re-viewed, especially if the backhaul network uses cellular or satellite communications.

CHAPTER-2

SECURING IOT

A BRIEF HISTORY OF OT SECURITY

- More than in most other sectors, cyber security incidents in industrial environments can result in physical consequences that can cause threats to human lives as well as damage to equipment, infrastructure, and the environment.
- Cyber security incidents have caused majority damage on the OT, For example, Stuxnet is thought to have been deployed on USB memory sticks up to two years before it was finally identified and discovered
- In addition to physical damage, operational interruptions have occurred in OT environments due to cyber security incidents
- For example, in 2000, the sewage control system of Maroochy Shire in Queensland, Australia, was accessed remotely, and it released 800,000 liters of sewage into the surrounding waterways.
- As technology has advanced, tools have been created to make attacks much easier to carry out.
- Many of the legacy protocols used in IoT environments are many decades old, and there was no thought of security when they were first developed.
- An important advantage for operators is the fact that they are far more familiar with their environment and have a better understanding of their processes, and can thus leverage multiple technologies and capabilities to defend their networks against attack.
- OT-specific communication systems have typically been standalone and physically isolated from the traditional IT enterprise networks in the same companies
- The isolation between industrial networks and the traditional IT business networks has been referred to as an “air gap,” suggesting that there are no links between the two.
- Broadly speaking, there is a varying amount of interconnection between OT and IT network environments, and many interdependencies between the two influence the level of interconnection.
- In addition to the policies, regulations, and governance imposed by the different industrial environments, there is also a certain amount of end-user preference and deployment-specific design that determines the degree of isolation between IT and OT environments.
- Evolution of ever-increasing IT technologies in the OT space comes with the benefits of increased accessibility and a larger base of skilled operators than with the nonstandard and proprietary communication methods in traditional industrial environments.
- The accessibility and scale makes security a major concern, particularly because many systems and devices in the operational domain were never envisioned to run on a shared,

open standards–based infrastructure, and they were not designed and developed with high levels of built-in security capabilities.

- Projects in industrial environments are often capital intensive, with an expected life span that can be measured in decades. The deployed OT systems often have slower development and upgrade cycles and can quickly become out of sync with traditional IT network environments.
- The proprietary nature of OT systems meant that threats from the outside world were unlikely to occur and were rarely addressed. There has, however, been a growing trend whereby OT system vulnerabilities have been exposed and reported.

COMMON CHALLENGES IN OT SECURITY

The security challenges faced in IoT are by no means new and are not limited to specific industrial environments. Some of the common challenges faced in IoT are explained below.

❖ **Erosion of Network Architecture**

Two of the major challenges in securing industrial environments have been initial design and ongoing maintenance. The initial design challenges arose from the concept that networks were safe due to physical separation from the enterprise with minimal or no connectivity to the out-side world, and the assumption that attackers lacked sufficient knowledge to carry out security attacks. The challenge, and the biggest threat to network security, is standards and best practices either being misunderstood or the network being poorly maintained. In many industries, the control systems consist of packages, skids, or components that are self-contained and may be integrated as semi-autonomous portions of the network. These packages may not be as fully or tightly integrated into the overall control system, network management tools, or security applications, resulting in potential risk.

❖ **Pervasive Legacy Systems**

Due to the static nature and long lifecycles of equipment in industrial environments, many operational systems may be deemed legacy systems. Beyond the endpoints, the communication infrastructure and shared centralized compute resources are often not built to comply with modern standards

❖ **Insecure Operational Protocols**

Many industrial control protocols, particularly those that are serial based, were designed without inherent strong security requirements. Common industrial protocols and their respective security concerns are discussed below

- **Modbus:** Modbus is commonly found in many industries, such as utilities and manufacturing environments, and has multiple variants. The security challenges that have existed with Modbus are not unusual. Authentication of communicating endpoints was not a default operation because it would allow an inappropriate source to send improper commands to the recipient

- **DNP3 (Distributed Network Protocol):** DNP3 is found in multiple deployment scenarios and industries. It is common in utilities and is also found in discrete and continuous process systems. There is an explicit “secure” version of DNP3, but there also remain many insecure implementations of DNP3 as well.
- **ICCP (Inter-Control Center Communications Protocol):** ICCP is a common control protocol in utilities across North America that is frequently used to communicate between utilities. ICCP was designed from inception to work across a WAN. Initial versions of ICCP had several significant gaps in the area of security such as no authentication for communication and encryption across the protocol was not enabled.
- **OPC (OLE for Process Control):** OPC is based on the Microsoft interoperability methodology Object Linking and Embedding (OLE). This is an example where an IT standard used within the IT domain and personal computers has been leveraged for use as a control protocol across an industrial network. Of particular concern with OPC is the dependence on the Remote Procedure Call (RPC) protocol, which creates two classes of exposure- vulnerabilities and level of risk.

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:** The Purdue Model for Control Hierarchy is the most widely used framework across industrial environments globally and is used in manufacturing, oil and gas, and many other industries.

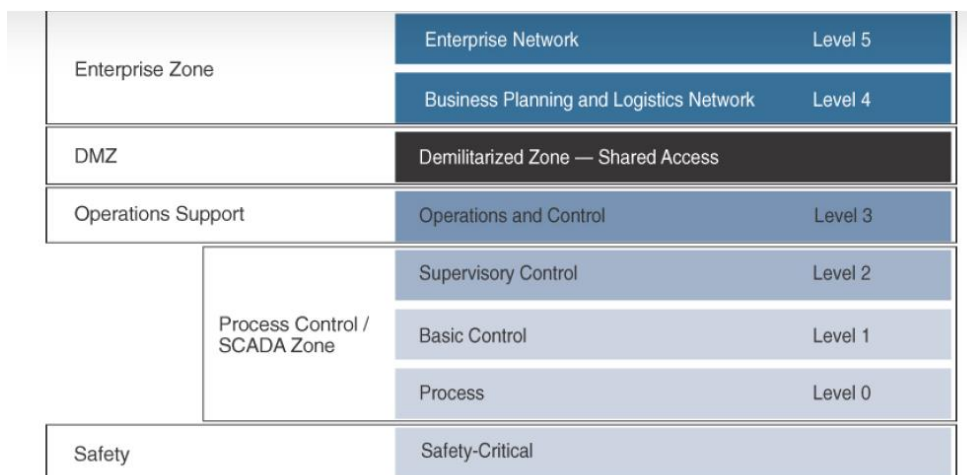


Figure: The Logical Framework Based on the Purdue Model for Control Hierarchy

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):

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.

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. DMZ resides between the IT and OT levels. Clearly, to protect the lower industrial layers, security technologies such as firewalls, proxy servers, and IPSs should be used to ensure that only authorized connections from trusted sources on expected ports are being used

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, system wide 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.

FORMAL RISK ANALYSIS STRUCTURES: OCTAVE AND FAIR

OCTAVE (Operationally Critical Threat, Asset and Vulnerability Evaluation) is a standard for risk definition from the Software Engineering Institute at Carnegie Mellon University

FAIR (Factor Analysis of Information Risk) is a standard for risk definition from The Open Group

➤ OCTAVE

OCTAVE has undergone multiple iterations. OCTAVE Allegro is 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.

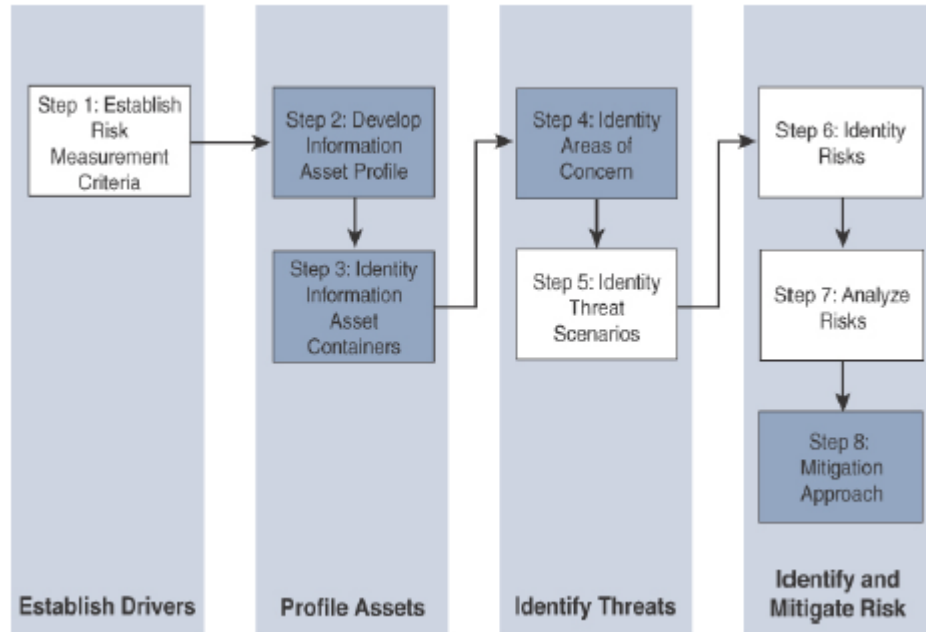


Figure: OCTAVE Allegro steps and phases

- ✓ The first step of the OCTAVE Allegro methodology is to establish a risk measurement criterion. OCTAVE provides a fairly simple means of doing this with an emphasis on impact, value, and measurement.
- ✓ The second step is to develop an information asset profile. This profile is populated with assets, a prioritization of assets, attributes associated with each asset, including owners, custodians, people, explicit security requirements, and technology assets.
- ✓ The third step is to identify information asset containers. Roughly speaking, this is the range of transports and possible locations where the information might reside. This references the compute elements and the networks by which they communicate. However, it can also mean physical manifestations such as hard copy documents or even the people who know the information.
- ✓ The fourth step is to identify areas of concern. At this stage, the analyst looks to risk profiles and delves into the previously mentioned risk analysis. It is no longer just facts, but there is also an element of creativity that can factor into the evaluation. History both within and outside the organization can contribute. References to similar operational use cases and incidents of security failures are reasonable associations.
- ✓ Closely related is the fifth step, where threat scenarios are identified. Threats are broadly (and properly) identified as potential undesirable events. It is at this point that an explicit identification of actors, motives, and outcomes occurs.

- ✓ At the sixth step risks are identified. Within OCTAVE, risk is the possibility of an undesired outcome. This is extended to focus on how the organization is impacted.
- ✓ The seventh step is risk analysis, with the effort placed on qualitative evaluation of the impacts of the risk.
- ✓ Finally, mitigation is applied at the eighth step. There are three outputs or decisions to be taken at this stage. One may be to accept a risk and do nothing, other than document the situation, potential outcomes, and reasons for accepting the risk. The second is to mitigate the risk with whatever control effort is required. The final possible action is to defer a decision, meaning risk is neither accepted nor mitigated.

➤ **FAIR**

- ✓ FAIR places emphasis on both unambiguous definitions and the idea that risk and associated attributes are measurable.
- ✓ 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
- ✓ The other side of the risk taxonomy is the probable loss magnitude (PLM), which begins to quantify the impacts, with the emphasis again being on measurable metrics.
- ✓ FAIR defines six forms of loss, four of them externally focused and two internally focused. Of particular value for operational teams are productivity and replacement loss. Response loss is also reasonably measured, with fines and judgments easy to measure but difficult to predict. Finally, competitive advantage and reputation are the least measurable.

INTRODUCTION TO ARDUINO

Arduino is a tiny computer that can be programmed to read information from the world around us and to send commands to outside world.

Arduino can be connected to several devices and electrical circuits.

The brain of the Arduino uno is an ATmega328p (microcontroller) chip where the program is stored. Instruction to the microcontroller are given by using Arduino programming language(c,c++). Arduino software (IDE-integrated development environment) is used for development.

Why Arduino?

- Arduino is an open source product, software/hardware which is accessible and flexible to customers
- Arduino is flexible because it has variety of digital and analog pins, SPI and PWM outputs.
- Arduino is easy to use, connected to a computer through a USB and communicates using serial protocol.
- Inexpensive, around 500rupees per board with free editable software.
- Arduino has growing online community where lots of source code is available for use, share and post examples for others to use.
- Arduino is cross platform, which can work on windows, Mac or Linux platforms.
- Arduino follows simple, clear programming environment as C language.

Which Arduino?

Hundreds of Arduino boards are available in the market. Among which the popular Arduino Uno is used in almost 99% of projects.

Some of the boards from Arduino family are

- Arduino Mega has more memory and pins with ATmega2560 chip. This is useful where Arduino Uno is not sufficient for the project.

- Arduino Micro is a bit smaller with a chip ATmega32u4 that can act like a keyboard or mouse which does its task with a 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. This is useful in internet of things projects.
- Flora is an Arduino compatible from Adafruit which is round wearable which can be sewed into clothing

EXPLORING ARDUINO UNO LEARNING BOARD

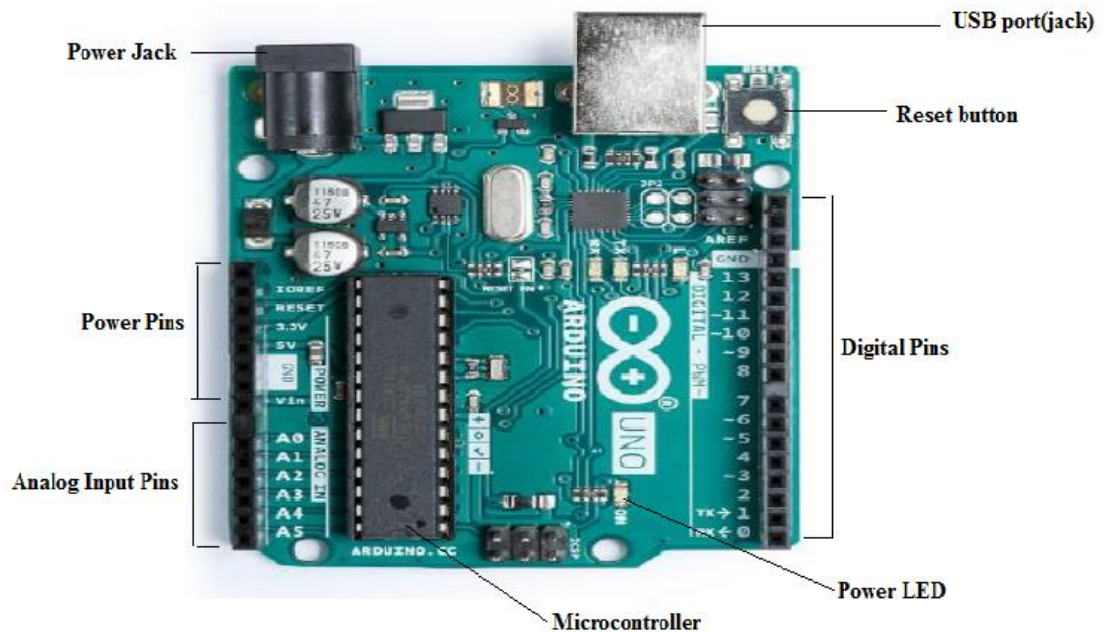


Figure: Arduino UNO Learning Board

- **Microcontroller:** the ATmega328p is the Arduino brain. Everything on the Arduino board support this microcontroller.
- **Digital pins:** Arduino has 14 digital pins labeled from 0 to 13 that can act as inputs or outputs.
 - When set as inputs these pins can read voltage(HIGH, LOW). When set as outputs these pins can apply voltages(5V→HIGH, 0V→ LOW)

- **PWM pins:** these are digital pins marked with ~ (pins 11,10,9,6,5, and 3). Pulse width modulation(PWM) pins allows to make digital pins output fake varying amounts of voltage.
- **TX and RX pins:** Digital pins 0 and 1. The T stands for transmit and R for Receive. Arduino uses these pins to communicate with the computer. These pins should be used in case of shortage of pins for projects else t has to be avoided.
- **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 A5and are most often used to read analog sensors. They can read different amounts of voltage between 0 and 5V. these can also be used as digital output/input pins like digital pins.
- **Power Pins:** The Arduino has 3.3V nd5V supply. The pin labeled as GND is the ground pin.
- **Reset Button:** when this button is pressed the program that is currently running on Arduino will start from beginning. There is a reset pin next to power pin. When a small voltage is applied to this pin it will reset Arduino.
- **Power ON LED:** will be on when power is applied to the Arduino.
- **USB jack:** By connecting a cable to this jack program is uploaded.
- **Power Jack:** This jack is used to power up Arduino.

Things that Arduino Can Do

- Control LED
- Display a message in a display like an LCD display
- Control DC or Servo motors
- Read data from outside world
- Motion Sensor: allows detecting movement.
- Light Sensor: allows to measure the quantity of light in outside world.
- Humidity and temperature sensor: measures humidity and temperature.
- Ultrasonic 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 Arduino.

INSTALLING THE SOFTWARE(ARDUINO IDE)

The Arduino Software (allows to write programs and upload them to board. In the Arduino Software page you will find two options

- 1 **Online IDE (Arduino Web Editor)** :It will allow to save sketches in the cloud, having them available from any device and backed up
- 2 **Offline** should use the latest version of the desktop IDE

Install the Arduino Desktop IDE accordingly to operating system

- Windows
- Mac OS X
- Linux
- Portable IDE (Windows and Linux)

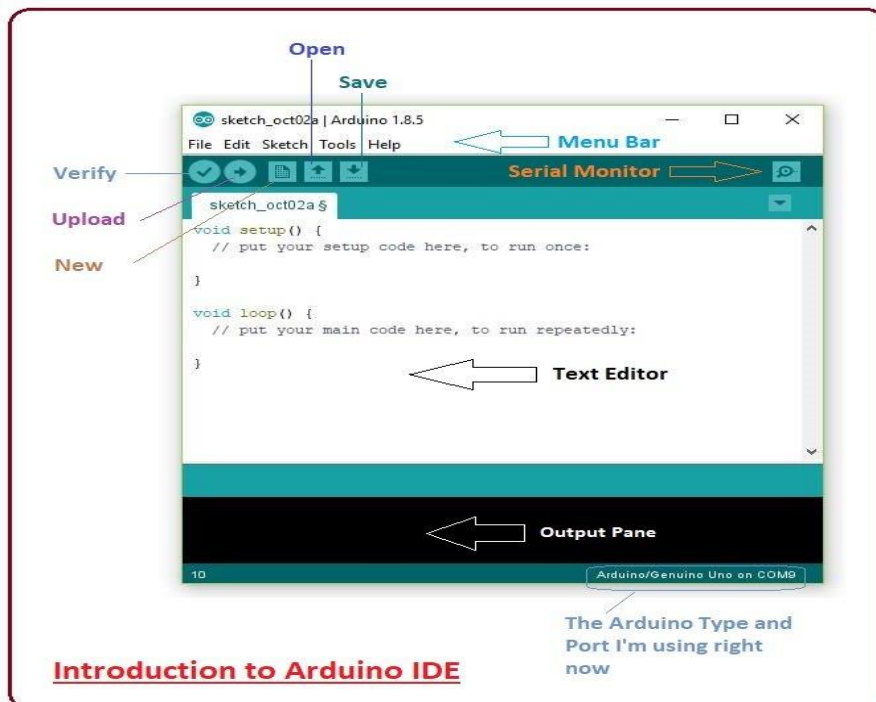


Figure: Arduino IDE

The toolbar buttons and functions of each button are as shown in the table.

Verify/Compile	Checks the code for errors
Stop	Stop the serial monitor or un-highlight other buttons
New	Creates a new blank sketch. Enter a name and location for your sketch
Open	Shows a list of sketches in your sketchbook
Upload	Uploads the current sketch to the Arduino. You need to make sure that you have the current board and port selected(in tools menu) before uploading
Serial monitor	Display serial data being sent from Arduino
Verify/compile	Button is used to check that your code is correct before you upload it to your Arduino
Stop button	Will stop the serial monitor from operating. If you need to obtain a snapshot of the serial data to be examined.

Table: Toolbar options in Arduino IDE

Technical Specifications of Arduino Uno is listed in the below table

Microcontroller n Arduino UNO	ATmega328p
Operating Voltage	5V
Input Voltage(recommended)	7-12V
Input voltage(limit)	6-20V
Digital I/O pins	14(of which 6 provide PWM outputs)
PWM Digital I/O pins	6
Analog input pins	6
DC Current pr I/O pin	20mA
DC Current for 3.3V pin	50mA
Flash Memory	32KB(ATmega328p) of which 0.5 KB used by bootloader
SRAM	2KB(ATmega328p)
EEPROM	1KB(ATmega328p)
Clock Speed	16MHz

Table: Technical Specifications of Arduino UNO

Connecting Arduino Uno Learning Board

- If you want to program your Arduino Uno while offline you need to install the Arduino Desktop IDE
- Connect your Uno board with an A B USB cable sometimes this cable is called a USB printer cable
- If you used the Installer, Windows from XP up to 10 will install drivers automatically as soon as you connect your board.
- You'll need to select the entry in the Tools Board menu that corresponds to your Arduino or Genuino board as shown in the figure

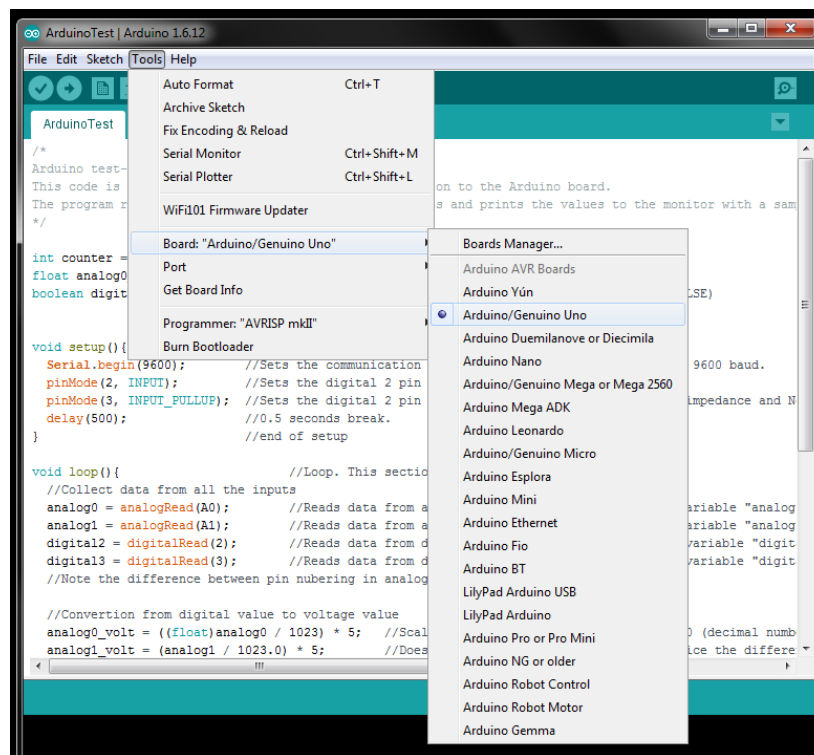


Figure: Selecting the right board

- Select the serial device of the board from the Tools Serial Port menu This is likely to be COM 3 or higher (COM 1 and COM 2 are usually reserved for hardware serial ports) To find out, you can disconnect your board and re open the menu the entry that disappears should be the Arduino or Genuino board. Reconnect the board and select that serial port as shown n figure.

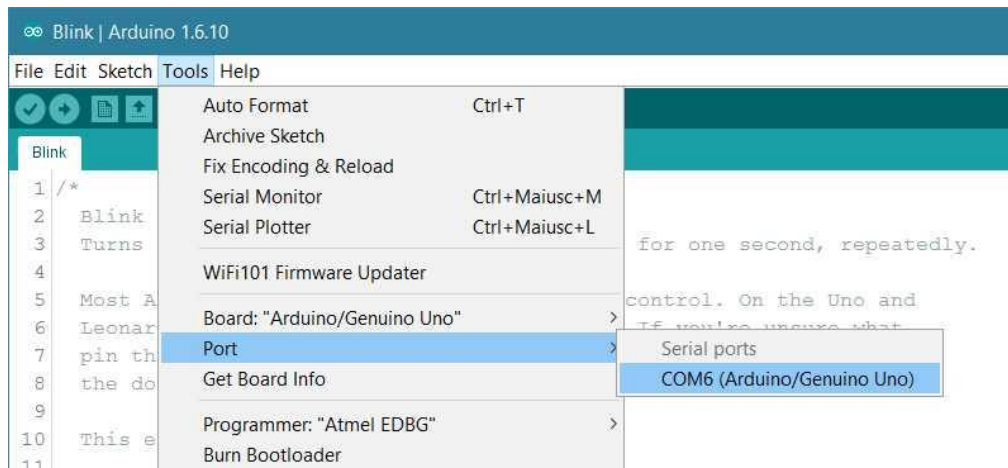
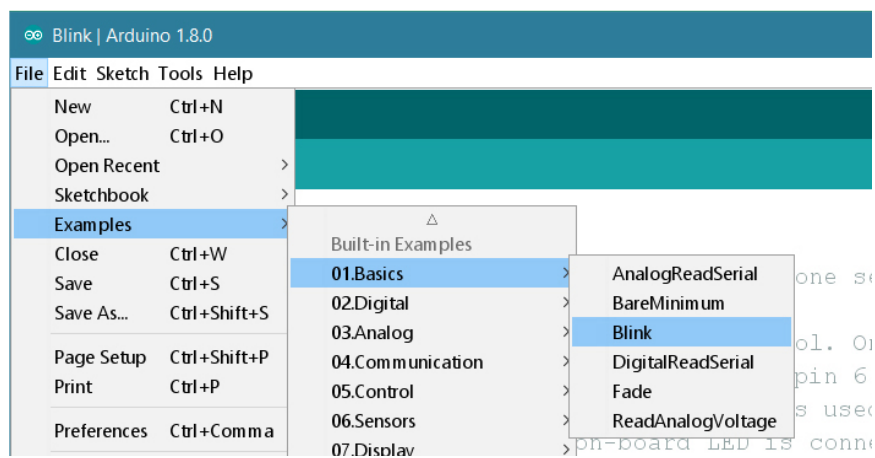
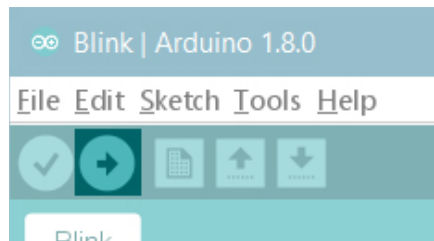


Figure: showing the layout of Arduino IDE selecting right port

- Open your first sketch
- Open the LED blink example sketch: File > Examples>01.Basics > Blink



- Upload the program
- Now, simply click the "Upload" button in the environment. Wait a few seconds you should see the RX and TX LEDs on the board flashing. If the upload is successful, the message "Done uploading" will appear in the status bar.



- A few seconds after the upload finishes, you should see the pin 13 (LED on the board) start to blink.

FUNDAMENTALS OF ARDUINO PROGRAMMING

The basic structure of Arduino programming with respect to usage of variables, constants, control flow statements and predefined functions to read the analog and digital inputs is as follows

Structure	<p>The structure contains two parts</p> <pre>void setup() //Preparation function used to declare variables // First function that runs only one in the program { Statement(s); //used to setup pins for serial communication } void loop() // execution block where instructions are executed repeatedly { Statement(s); //this is the core of the Arduino program // Functions include reading inputs triggering outputs etc }</pre>
void setup()	<pre>void setup() { pinMode(pin,INPUT); //’pin’ configured as input }</pre>
void loop()	<pre>void loop() //after calling setup(),loop() functions does its task { digitalWrite(pin,HIGH); //sets ‘pin’ ON delay(10000); // pause for 10000 milli seconds digitalWrite(pin, LOW) //sets ‘pin’ OFF delay(10000); // pause for 10000 milli seconds }</pre>
Functions	<p>A function is a piece of code that has a name and a set of statements executed when function is called. Functions are declared by its type followed with name of function.</p> <p>Syntax:</p> <pre>type functionName(parameters) { Statement(s); }</pre> <p>Example:</p> <pre>int delayvar() {</pre>

	<pre>int var; // create temporary variable var var=analogRead(potent); // read from potentiometer var=var/4; //convert the value of variable var return var; }</pre>																		
{ } curly braces	They define beginning and end of function blocks, unbalanced braces lead to error																		
Semicolon	It is used to end a statement and separate elements of a program.																		
/*...*/ block comments	Multiline comments begin with /* with a description of blocks and ends with */																		
// line comments	Single line comment begins with // and ends with next instruction followed																		
Variables	<p>A variable is a way of storing value for later use in the program. A variable type maybe int,long,float etc A variable can be defined and assigned an initial value A global variable is declared at the beginning of the program before setup() and can be used in any part of the program. A local variable is defined inside the function in which it is declared</p> <p>Example:</p> <pre>int var; // var is a global variable and can be used by all functios void setup() { } void loop() { for(int x=0;x<5;) { x++; // variable x can be used within the loop } float y; // variable y is can be used only in loop function }</pre>																		
Data Types	<table border="1"> <thead> <tr> <th>Data Type</th> <th>Syntax</th> <th>Range</th> </tr> </thead> <tbody> <tr> <td>Byte</td> <td>byte x=100;</td> <td>0-255</td> </tr> <tr> <td>Int</td> <td>int y=200;</td> <td>32767 to -32768</td> </tr> <tr> <td>Long</td> <td>long var = 8000;</td> <td>2147483647 to -2147483648</td> </tr> <tr> <td>Float</td> <td>float x= 3.14;</td> <td>3.4028235E+38 to -3.4028235E+38</td> </tr> <tr> <td>Arrays</td> <td>int myarray []={10,20,30}</td> <td>Size depends on the data type associated with declaration</td> </tr> </tbody> </table>	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+38 to -3.4028235E+38	Arrays	int myarray []={10,20,30}	Size depends on the data type associated with declaration
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+38 to -3.4028235E+38																	
Arrays	int myarray []={10,20,30}	Size depends on the data type associated with declaration																	

Operators	Operators		Syntax and usage								
	Arithmetic operators (+, -, /, *)		x=x+5; y=y-6; z=z*2; p=p/q;								
	Assignment operators (=, ++, --, +=, *=, /=)		x++; // same as x=x+1 x+=y; // same as x=x+y x-=y; // same as x=x-y x*=y; // same as x=x*y x/=y; // same as x=x/y								
	Comparison operators (==, !=, <, >, <=, >=)		x==y // x is equal to y x!=y // x is not equal to y x<y // x is less than y x!>y // x is no equal to y								
	Logical operators (&&, , !)		x>2 && x<5 // Evaluates to true only if both expression are true x>2 y>2 // Evaluates to true only if any one expression are true !x>2 // true if only expression is false								
	Constants		<table border="1"> <thead> <tr> <th>Constants</th> <th>Usage</th> </tr> </thead> <tbody> <tr> <td>TRUE/FALSE</td> <td> Boolean constants true=1 an false=0 defined in logic levels if(b==TRUE) { //do something } </td> </tr> <tr> <td>INPUT/ OUTPUT</td> <td>Used with pinMode () function to define levels. pinMode(13,OUTPUT)</td> </tr> <tr> <td>HIGH/LOW</td> <td>Used to define pin levels HIGH→1 ON, 5 volts LOW→0, OFF, 0 volts Digital Write(13,HIGH)</td> </tr> </tbody> </table>		Constants	Usage	TRUE/FALSE	Boolean constants true=1 an 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
Constants	Usage										
TRUE/FALSE	Boolean constants true=1 an 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 comparisons results in a true value }
if..else	if(input==HIGH)

	<pre> { Statement(s); //Evaluated only if comparison results in a true value } Else { Statement(s); // Evaluated only if comparison results in a false value } </pre>	
for	<pre> 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 } </pre>	
while	<p>While loop executes until the expression inside parenthesis becomes false</p> <pre> while(some_variable??value) { Statement(s); //Evaluated till comparison results in a false value } </pre>	
do....while	<p>Bottom evaluated loop, works same way as while loop but condition is tested at the end of loop.</p> <pre> Do { Dosomething; }while(somevalue); </pre>	
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
	DigitalRead(pin)	Read value from a specified pin with result being HIGH/LOW val=digital Read(pin); //Val will be equal to input pin
	Example	int x=13; //connect 'x' to 13 int p=7; //connect push button to pin 7

		<pre>int val=(); //variable to store the read value void setup() { pinMode(x, OUTPUT); //sets 'x' as OUTPUT } void loop() { val=digitalRead(p); //sets 'value' to 0 digitalWrite(x,val); //sets 'x' to button value }</pre>
Analog i/o	Methods	Usage
	analogRead(pin)	Reads value from a specified analog pin works on pins 0-5 val=analogRead(pin); //'val' equal to pin
	analogWrite(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	<pre>int x=10; //connect 'x' to pin 13 int p=0; pin 7 int val; //variable for reading void setup() { } // No setup is needed Void loop() { Val=analogRead(p); //sets 'value' to 0 Val/=4; analogWrite(x,val); //outputs PWM signal to 'x' }</pre>
time	Methods	Usage
	delay(ms)	Pauses for amount of time specified in milliseconds. delay(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 10 or 10
random	Methods	Usage
	random reed (value)	Sets a value / seed as a starting point for function

	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 to 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 analogWrite(x,number); //outputs PWM signal delay(500); }
Serial	Methods	Usage
	Serial.begin(rate)	Opens serial port and sets then baud rate for serial data transmission Void setup() { Serial.begin(9600); //sets default rate to 9600bps }
	Serial.println(data)	Prints data to the serial port Serial.println(value); //sends the 'value' to serial monitor

Differences between Analog, Digital and PWM pins

In analog pins, possible states are between 0 to 1023. This allows the users to read sensor values. For example, with a light sensor, if it is very dark the reading will be 1023, if it is very bright the reading will be 0. If there is a brightness between dark and bright the reading will have a value between 0 and 1023.

In digital pins there are only two possible states, which are on or off. These can also be referred as high or low, 1 or 0 and 5v or 0v. For example, if an LED is on, then its state is High or 1 or 5v. If it is off, the values will be Low or 0 or 0v.

PWM pins are digital pins, so they output either 0 or 5v. however these pins can output "fake" intermediate voltage values between 0 and 5v, because they can perform "pulse width modulation"(PWM). PWM allows to "simulate" varying levels of power by oscillating the output voltage o arduino.

INTRODUCTION TO RASPBERRY PI

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.

The later models got popular and are used for various applications such as robotics. Several generations of RaspberryPi are released. Some of the specifications of models are tabulated below.

RaspberryPi	Model A+	Model B	Model B+	2,Model B	Model 3
Quick Summary	Cheapest, smallest single board computer	The original RaspberryPi	More USB and GPIO that the B.Ideal choice for schools	Most advanced Raspberry Pi	Newest with wireless connectivity
Chip	Broadcom BCM 2835			Broadcom BCM 2836	Broadcom BCM2837
Processor	ARMv6 single core			ARMv7 quad core	4xARM cortex-A53
Processor Speed	700MHz			900MHz	1.2GHz
Voltage and Power draw	600mA @5V			650mA @5V	
GPU	Dual core video core IV Multimedia Co-Processor				Broadcom Video core IV
Size	65x56mm	85x56mm			
Memory	256MB SDRAM @ 400 MHz	512 MB SDRAM @400MHz		1GB SDRAM @ 400MHz	1GB LPDDR2(900MHz)
Storage	Micro SD card	SD card	Micro SD card		
GPIO	40	26	40		
USB 2.0	1	2	4		
Ethernet	None	10/100mb Ethernet RJ45Jack			
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 system	Raspbian RaspBMC, Arch Linux,rise OS, OpenEL EC Pidora				
Video output	HDMI Composite				
Supported Resolutions	640x350 to 1920x1200, including 1080p, PAL &NTSC standards				
Power Source	Micro USB				

Table: Technical Specification of Raspberry Pi Models

The foundation provides Raspbian, a Debian based Linux distribution for download as well as third party ubuntu, windows 10 IOT core, RISC OS and specialized media center distributions.

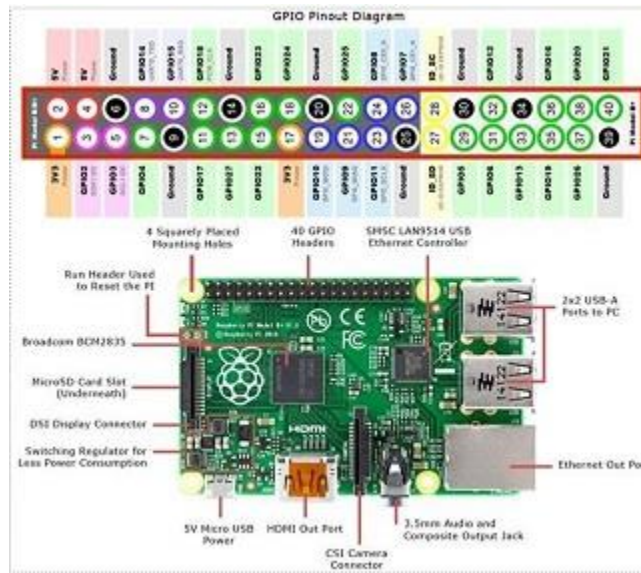
Foundation also provides python and scratch as the main programming language with support for other languages.

Why RaspberryPi?

- Inexpensive
- Cross-platform
- Simple clear programming environment
- Open source and extensible software
- Open source and extensible hardware

Exploring RaspberryPi Learning Board

The figure shows the Raspberry Pi board labeled.



- ❖ **Processor:** The Broadcom BCM2835 Soc is the first generation RaspberryPi. These chips are equivalent to the chips used in smart phones. The RaspberryPi 2 uses a Broadcom BCM2836 Soc with a 900MHz 32 bit quad core ARM cortex A7 processor with 256KB shared L2 cache.
- ❖ **Power Source:** The easiest way to power the RaspberryPi is through the micro USB port on the side of the unit. The recommended input voltage is 5v and input current is 2A. The Raspberry Pi can operate on lower power supplies such as 5v @ 1A. Any excessive use of the use of the USB ports or heavy CPU loading can cause the voltage drop and instability during use.
- ❖ **SD card:** The working framework is stacked on a SD card space on the RaspberryPi . There are no locally available storage accessible for RaspberryPi.
- ❖ **GPIO(General Purpose input Output):** These are non-specific pins on a co-ordinated circuit to know an input or output pin. These can be controlled by the client. GPIO capabilities may include
 - ✓ GPIO pins can be designed to be input or output
 - ✓ GPIO pins can be empowered/crippled
 - ✓ Input values are meaningful(normally high=1, low=0)
 - ✓ Yield values are writable/meaningful
 - ✓ Input values can frequently be utilized as IRQs(regularly for wakeup occasions)

In programming environment the GPIO.BOARD points to the pins by the number of the pin. The GPIO.BCM points to the pins by the “Broadcom SOC channel” number; these are the numbers after “GPIO” in the green rectangles around the outside of the underneath graphs

- ❖ **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. The mobile industry processor interface(MIPI) inside the Broadcom BCM2835 IC feeds graphics data directly to the display panel through this connector.
- ❖ **Audio Jack:** A standard 3.5mm TRS connector is accessible on the RPi for stereo sound yield. Any earphone or 3.5mm sound link can be associated straightforwardly. In spite of the fact that jack can’t be utilized for taking sound information, USB mics or USB sound cards can be utilized.
- ❖ **Status LEDs:** There are 5 status LEDs on the RPi that demonstrate the status of different exercises such as:
 - ✓ OK- SD card Access⁹by means of GPIO16)-named as “OK” on Model B Rev1.0 sheets and “ACT” on Model B rev2.0 and Model A sheets.
 - ✓ POWER- 3.3 power- named as “PWR” on all the boards
 - ✓ FDX-Full Duplex(LAN) (Model B)-marked as “FDX” on all the boards
 - ✓ LNK-Link/Activity(LAN)(Model B)-marked as “LNK” on all the boards
 - ✓ 10M/100-10/100Mbit(LAN)(Model B)-named (erroneously) as “10M” on Model B Rev 1.0 boards and “100” on model B Rev 2.0 and Model A boards USB ports.

There is 1 port on Model A, 2 on Model B and 4 on Model B+ operates at a current upto 100mA, An external USB powered hub is required to draw current more than 100mA.

- ❖ **Ethernet Port:** Ethernet port is accessible on Model B and B+. It can be associated with a system or web utilizing a standard LAN link on Ethernet port. The Ethernet port are controlled by Microchip LAN9512 LAN controller chip.
- ❖ **CSI connector(CSI):** Camera Serial interface is a serial interface outlined by MIPI(Mobile Industry Processor Interface?) organization together went for interfacing computerized camera with a portable processor. The RPi establishment gives a camera uncommonly made to the Pi which can be associated with the Pi utilizing the CSI connector.
- ❖ **JTAG(Joint Test Action Group) headers:** JTAG association was started in mid 1980s. This association was started to address test point get to issues on PCB with surface mount gadgets. The association formulated a technique for access to gadget pins by means of a serial port that got be distinctly known as the TAP(Test Access Port). In 1990 the strategy was turned into a universal standard(IEE std 1149.1). A large number of gadgets now use this institutionalized port as a component to permit test and configuration architects to get to pins.
- ❖ **HDMI:** High definition Multimedia interface to give both video and sound yield

RASPBERRYPI OPERATING SYSTEMS

- Various operating systems can be installed on RaspberryPi through SD cards. Most use a MicroSD slot located on the bottom of the board
 - The RaspberyPi primarily uses Raspbian, a debian based Linux operating system
 - Other third party operating systems available via the official website include Ubuntu MATE, Snappy Ubuntu core, Windows 10 IoT Core, RISC OS and specilaized distribution for the Kodi media center and classroom management.
- **Operating systems (not Linux based)**
- ✓ RISC OS Pi(a special cut down version RISC OS Pico for 16MB cards and large for all models of Pi 1 and 2 has also been made available)
 - ✓ FreeBSD
 - ✓ NetBSD
 - ✓ Plan 9 from Bell Lbs 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 implementation of sixth edition Unix OS for teaching purpose, it is ported to RaspberryPi from MIT Xv6, which can boot NOOBs
 - ✓ Haiku-is an open source BeOS clone that can compile for the RaspberryPi and several other ARM boards.

➤ **Operating systems (Linux based)**

- ✓ Xbian-sing Kodi open source digital media center
- ✓ openSUSE
- ✓ RaspberryPi Fedora remix
- ✓ Pidora another Fedora remix optimized for the raspberryPi
- ✓ Gentoo Linux
- ✓ Diet Pi
- ✓ CentOS\openWrt
- ✓ Kali Linux
- ✓ Ark OS
- ✓ Kano OS
- ✓ Nard SDK

➤ **Media Center Operating systems**

- ✓ OSMC
- ✓ OpenELEC
- ✓ LibreELEC
- ✓ Xbian
- ✓ Raspdex

➤ **Audio Operating systems**

- ✓ Volumio
- ✓ Pimusicbox
- ✓ Runeaudio
- ✓ moOdeaudio

➤ **Recalbox**

- ✓ Happi Game Center
- ✓ Lakka
- ✓ ChameleonPi
- ✓ Piplay

PROGRAMMING RASPBERRY PI WITH PYTHON

Raspberry Pi runs Linux and supports python out of the box. The general purpose input/output capability provided by GPIO pins on Raspberry Pi makes it useful device for Internet of Things. Some of the simple python programs are tabulated below

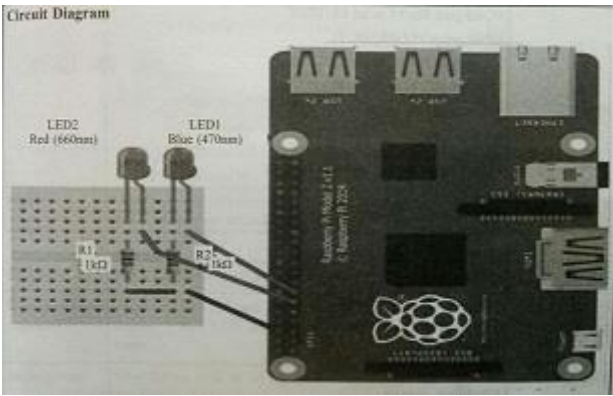
Program	Code
Print hello world	<pre>print("hello world")</pre>
Program to add two numbers	<pre>a=1.2 b=5.3 sum=float(a)+float(b) print("the sum of{0} and {1} is {2}".format(a,b,sum))</pre>
Program to roll a dice	<pre>import random min=1 max=6 roll_again="yes" while roll_again=="yes" or roll_again=="y" print("rolling the dices") print("the values are") print(random.randint(min,max)) print(random.randint(min,max))</pre>
Program to find the IP address of Raspberry Pi	<pre>import urllib import re print("we will try to open this url,in order to get ip address") url=http://checkup.dyndnd.org print(url)</pre>
Program to generate password	<pre>import string from random import* characters=string.ascii_letters+string.punctuation+string.digits password="".join(choice(characters)for x in range(randint(8,16))) print(password)</pre>
Program to generate fibnocci series	<pre>a,b=0,1 while b<200: print(b) a,b=b,a+b</pre>
Program to check for armstrong number	<pre>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")</pre>

	else: print(num, "is not an Armstrong number")
Program to display calendar of given month of the year	import calendar yy=2021 mm=07 print(calendar.month(yy,mm))

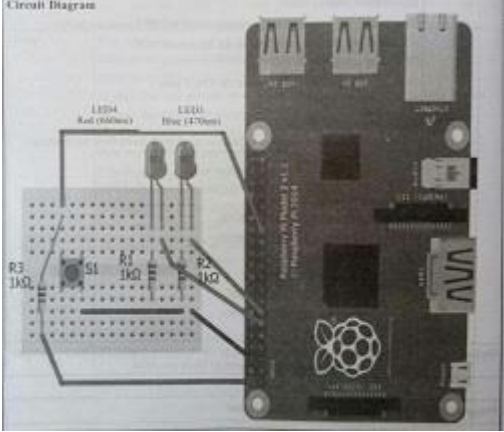
Table: Simple python programs on RaspberryPi

RaspberryPi can be interfaced with variety of sensors,actuators using GPIO pins and also SPI, I2C and serial interfaces. Input from the RaspberryPi can be processed and action can be taken for instance sending data to server, sending an email, triggering a relay switch. Some of the interfacing programs on RaspberryPi are tabulated below.

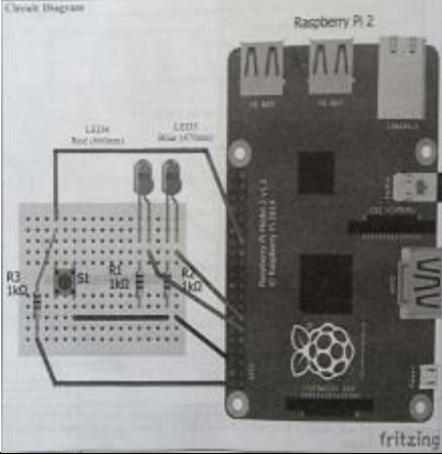
Program #1	Printing to a terminal
Components required	Raspberry Pi +SD card, monitor+HDMI cable, keyboard,mouse and power supply, breadboard
Description	Printing a message "hello world" using python programming. To print a greeting message to the console. The steps are listed below
Key points	<ol style="list-style-type: none"> 1. Find your Raspberry Pi 2. Mount 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 "raspberrry" 9. Double click on "terminal" 10. This will load the "terminal" 11. Type the following commands <ul style="list-style-type: none"> ✓ Change the directoryby the command \$ cd Desktop ✓ Create new directory \$ mkdir python_code ✓ Change directory to python code \$ cd python_code ✓ Create new file helloworld.py ✓ Now enter the given code which is given 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 print on the console

Program #2	Blinking an LED
Components required	Raspberry Pi +SD card, monitor+HDMI cable, keyboard, mouse and power supply, 1Red LED and Blue LED, two 1K resistors, breadboard
Description	Example of controlling the State of LED from ON to OFF and vice versa from Raspberry Pi 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 running the two programs given below
Circuit diagram	
Key points	<ol style="list-style-type: none"> 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	<pre> 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 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 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) </pre>

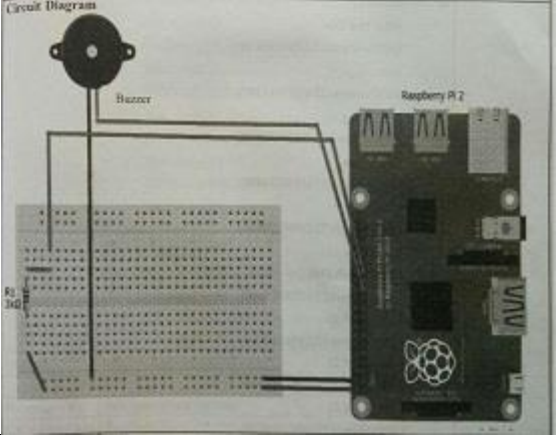
	<pre> #turn LEDs off pin 17 GPIO.output(17,GPIO.LOW) #turn LEDs off pin 27 GPIO.output(27,GPIO.LOW) #wait for 1 second time.sleep(1) File: blink_ever.py #Access the python working environment #!/usr/bin/python #import the time module so as to switch LEDs on/off with the time elapsed import time #import 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 as 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 LEDs off GPIO.output(17,GPIO.LOW) GPIO.output(27,GPIO.LOW) time.sleep(1) </pre>
Output	LEDs turns on/off twice when blink.py file is executed and LEDs keep 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, 1Red LED and Blue LED, two 1K resistors, push button and jumper wires, breadboard
Description	Example for controlling an LED with a switch. This example shows how to get input from GPIO pins and process the state of LED. In the infinite while loop the value of pin 10 is checked and stat of the LED is toggled if switch is pressed.
Circuit diagram	
Key points	<ol style="list-style-type: none"> 1. Create file <code>buton.py</code> 2. Enter the code 3. To run the python code <code>“sudo python.py”</code>
Code	<pre> File:buton.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 as 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 while True: if (GPIO.input(10)==True); print(“Button Pressed”) #print the time when the input was given from the push button </pre>

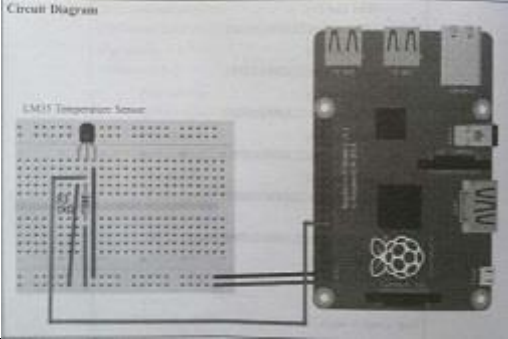
	<pre> os.system('date') #Read the status of a button from GPIO pin 10 print GPIO.input(10) #wait for 5 seconds time.sleep(5) else: #clear the system variables os.system('clear') #prompt the user to give an input print("waiting for you to press a button") time.sleep(1) </pre>
Output	Press the push button switch to turn ON/OFF LED

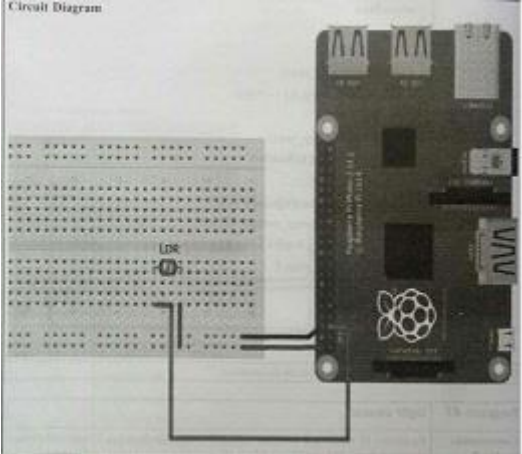
Program #4	Interact with the user
Components required	Raspberry Pi +SD card, monitor+HDMI cable, keyboard, mouse and power supply, 1Red LED and Blue LED, two 1K resistors, push button and jumper wires, breadboard
Description	Example showing how to get input from a user and process the state of the LEDs Example shows a python program for controlling the LED by asking the user to choose which LED would blink(option "1" for Red and option "2" for Blue) and how many times the LED would blink.
Circuit diagram	
Key points	<ol style="list-style-type: none"> 1. Create file "user_input.py" 2. Enter the code 3. Run the python file by "sudo python user_input.py"<<Run through the questions and make an LED blink
Code	<p>File: user_input.py</p> <pre> #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 </pre>

	<pre> GPIO.setmode(GPIO.BCM) #configure pin 17 and 27 as an output pins GPIO.setup(17,GPIO.OUT) 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 print "you picked the Red LED" counter= input("How many times would you like to blink?: ") while counter>0: #on LED on pin 17 GPIO.output(17,GPIO.HIGH) time.sleep(1) #off LED on pin 17 GPIO.output(17,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 print "you picked the Blue LED" counter=input("How many times would you like 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 </pre>
Output	LED gets flicked on the inputs given by the user

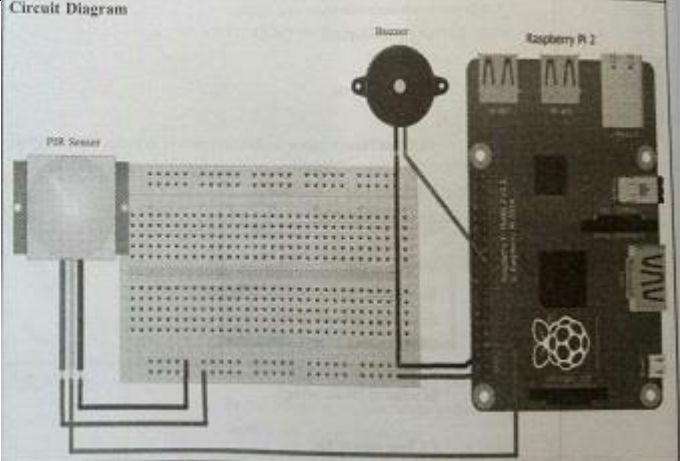
Program #5	BUZZER
Components required	Raspberry Pi +SD card, monitor+HDMI cable, keyboard, mouse and power supply, 1Red LED and Blue LED, two 1K resistors, push button and jumper wires, breadboard, buzzer
Description	Example of python program for controlling peizo buzzer by reading an input value which runs over a loop to beep number of times the user has chosen. Peizo buzzer is connected to pin 22 and switch to raspberry pi
Circuit diagram	
Key points	<ol style="list-style-type: none"> 1. Create file “buzzer.py” 2. Enter the code 3. To run python code “sudo python buzzer.py”<<listen to it beep
Code	<pre> 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 dash GPIO.output(22,GPIO.LOW) time.sleep(.2) GPIO.output(22,GPIO.HIGH) </pre>

	<pre> 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() </pre>
Output	Listen to beep of peizo buzzer

Program #6	Temperature Sensor
Components required	Raspberry Pi +SD card, monitor+HDMI cable, keyboard, mouse and power supply, 1Red LED and Blue LED, two 1K resistors, push button and jumper wires, breadboard, buzzer, LM35 temperature sensor
Description	
Circuit diagram	
Key points	
Code	<pre> import os import glob import time #initialise 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) </pre>
Output	Current room temperature is recorded

Program #7	Light Sensor
Components required	Raspberry Pi +SD card, monitor+HDMI cable, keyboard, mouse and power supply, 1Red LED and Blue LED, two 1K resistors, push button and jumper wires, breadboard, buzzer, LM35 temperature sensor, LDR light dependant sensor
Description	<p>An example involving an LDR sensor which reads the intensity of light and records it a text files.</p> <p>This example shows how to get an analog input from GPIO pins and process the input.</p> <p>An infinite loop runs over the sensor which records intensity of light with date and time stamps every second.</p>
Circuit diagram	
Key points	<ol style="list-style-type: none"> 1. Create file “ldr.py” an “touch foo.txt” 2. To run the python code “sudo python ldr.py”<<see what the light levels in the room are 3. Check the file “more foo.txt” to check the result
Code	<pre> 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, GPI.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: </pre>

	<pre> 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 opened file fo.close() time.sleep(1) </pre>
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, 1Red LED and Blue LED, two 1K resistors, push button and jumper wires, breadboard, buzzer, LM35 temperature sensor, LDR light dependant sensor,
Description	<p>An example involving an PIR sensor which detects the motion of an object. This example shows how to get an analog input from GPIO pins and process the input.</p> <p>An infinite loop runs over the PIR sensor which waits for any of the movements across its boundary.</p>
Circuit diagram	
Key points	<ol style="list-style-type: none"> 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	<pre> File:PIR.py #!/usr/bin/env python import RPi.GPIO as GPIO import time GPIO.setmode(GPIO.BCM) </pre>

	<pre> 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 eset GPIO.cleanup() </pre>
Output	Move in front of the PIR to activate it and sensor generates a message

CHAPTER-2

SMART AND CONNECTED CITIES

SMART CITY IOT ARCHITECTURE

A smart city IoT infrastructure is a four-layered architecture, as shown in Figure. 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.

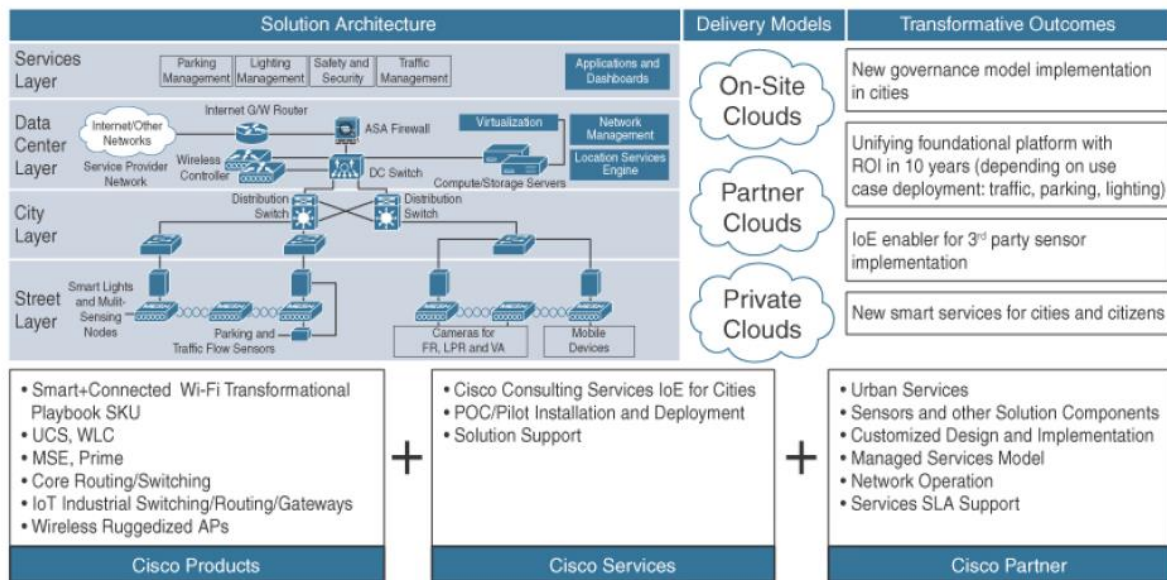


Figure: Smart city layered architecture

- **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. Sensor devices are able to detect and measure events in the physical world. A variety of sensors are used at the street layer for a variety of smart city use cases such as A magnetic sensor can detect a parking event, An air quality sensor, A lighting controller, Video cameras etc. The choice of sensor technology depends on the exact nature of the problem, the accuracy and cost trade-offs appropriate for it, and any installation limitations posed by the physical environment.
- **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. One key consideration of the city layer is that it needs to transport multiple types of protocols, for multiple types of IoT applications. The city layer must be built around resiliency, to ensure that a packet coming from a sensor or a gate-way will always be forwarded successfully to the headend station. Figure shows one such

approach. In this model, at least two paths exist from any aggregation switch to the data center layer. A common protocol used to ensure this resiliency is Resilient Ethernet Protocol (REP).

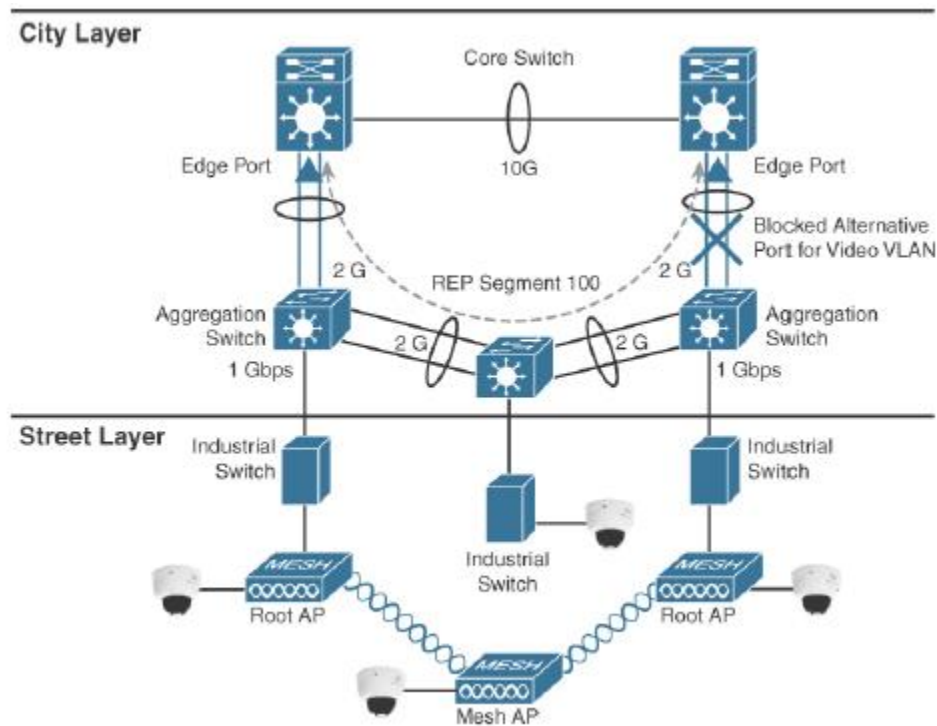


Figure: Street layer resiliency

- **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. 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. This proximity and flexibility also facilitate the ex-change of information between smart systems and allow for the deployment of new applications that can leverage information from several IoT systems. Figure 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. However, not all data is processed in the central cloud-based data center. Most of the real-time and locally significant data can be directly processed at the edge of the network, leveraging a fog architecture

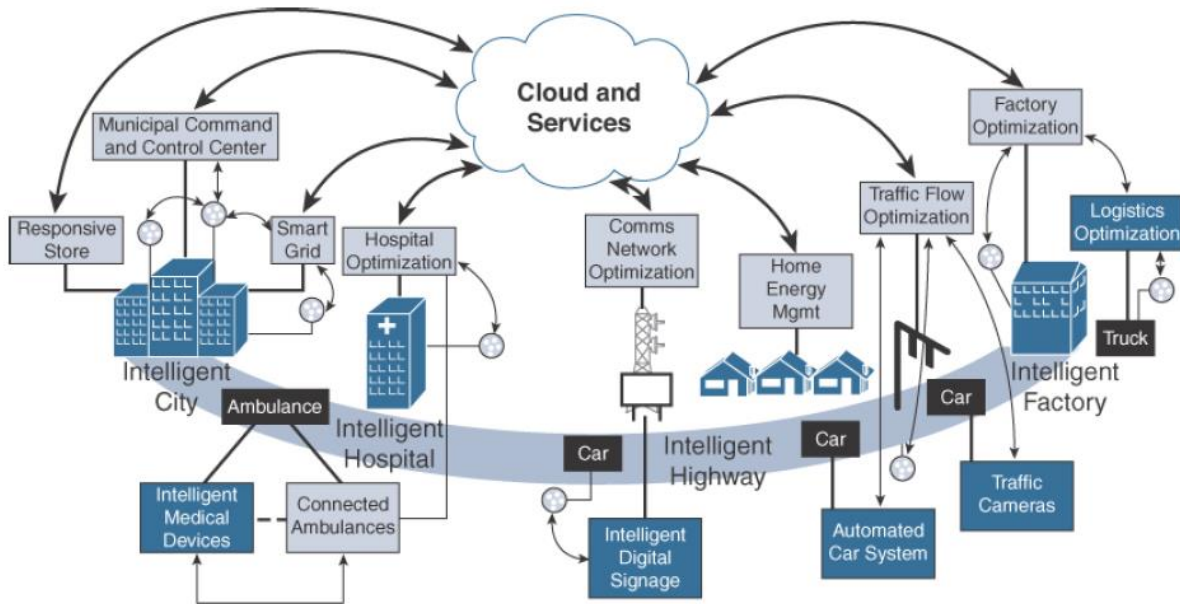


Figure: The role of cloud for smart city applications

- **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.

The architecture provides application developers and sensor vendors with the tools necessary to innovate and invent new community experiences via open APIs, software development kits (SDKs), city information models, and more to develop city-qualified applications that drive high-value smart city services.

SMART CITY SECURITY ARCHITECTURE

Security architecture for smart cities must utilize security protocols to fortify each layer of the architecture and protect city data. Figure shows reference architecture, with specific security elements high-lighted. Security protocols should authenticate the various components and protect data transport throughout. The security architecture should be able to evolve with the latest technology and incorporate regional guidelines. Network partners may also have their own compliance standards, security policies, and governance requirements that need to be added to the local city requirements.

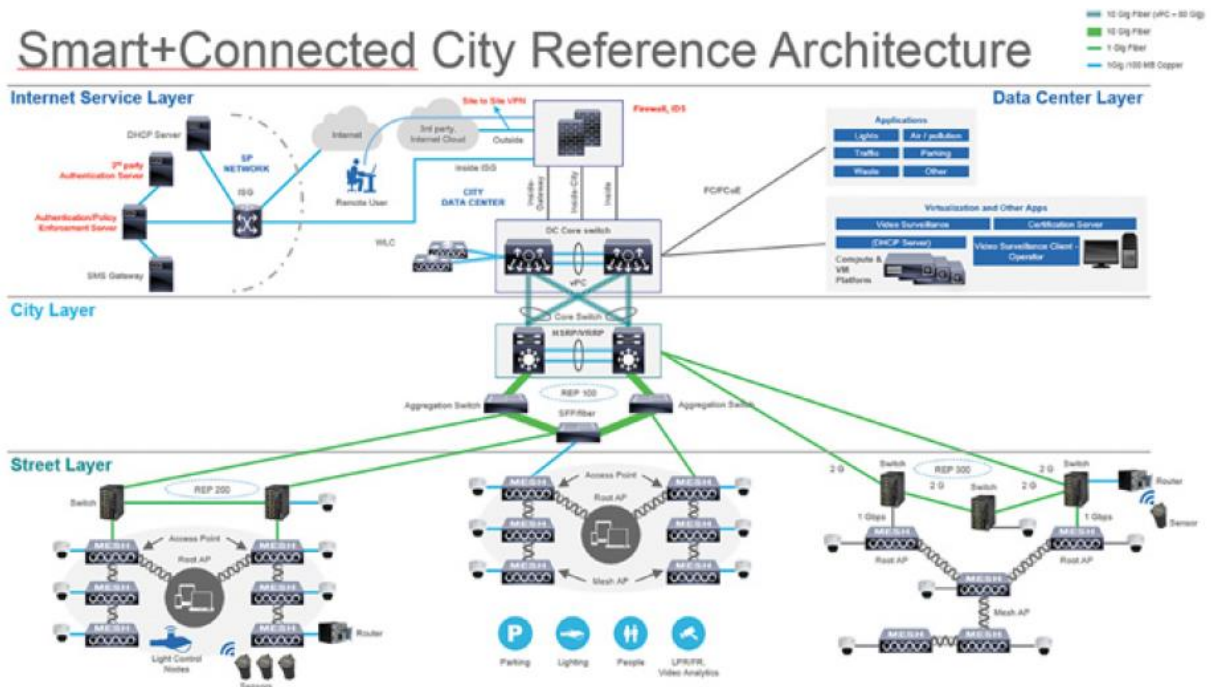


Figure: Key Smart and Connected Cities Reference Architecture

- Starting from the street level, sensors should have their own security protocols. Some industry-standard security features include device/sensor identification and authorization; device/sensor data encryption. Another consideration may be the type of data that the sensor is able to collect and process.
- Data should be secured both at rest and in motion, but when data is stored, additional security needs to be put in place to ensure that information will not be tampered with, abused, or stolen.
- The city layer transports data between the street layer and the data center layer. It acts as the network layer.
- 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.

SMART CITY USE-CASE EXAMPLES

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**

Street lighting comprises one of the largest expenses in a municipality's utility bill. 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. 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.

→**Street Lighting Architecture**

Connected lighting uses a light management application to manage streetlights 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. 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

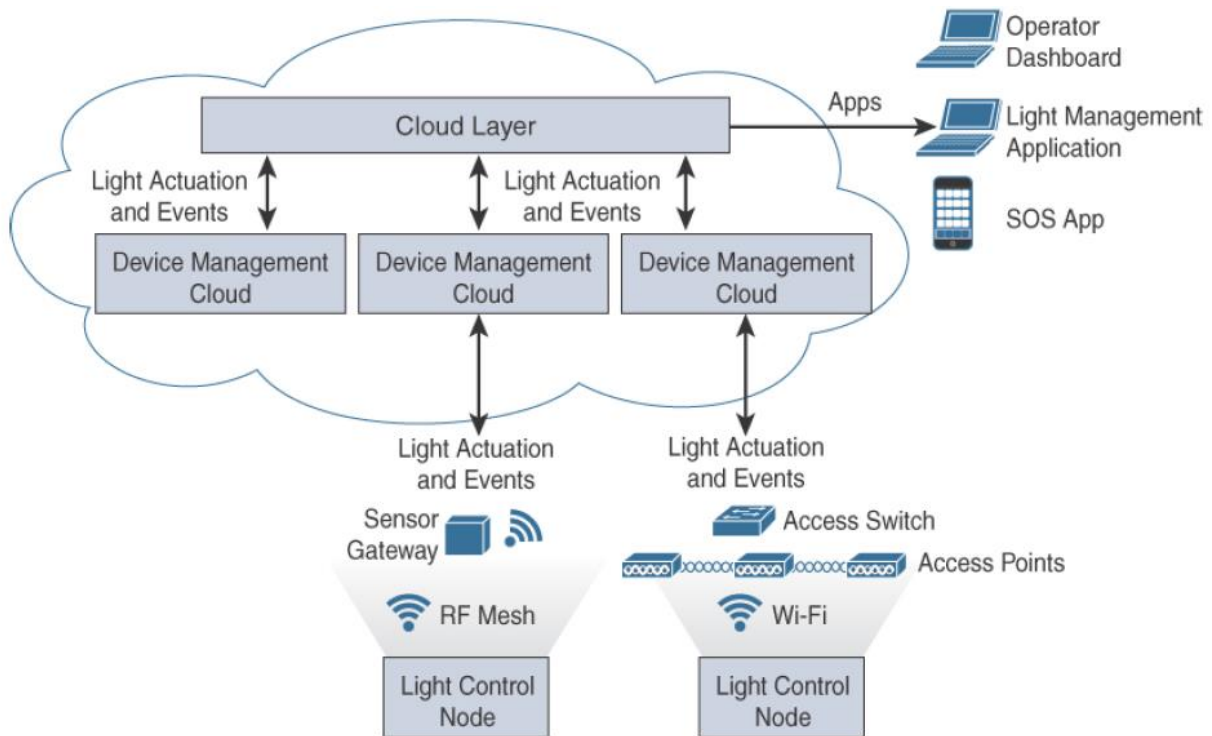


Figure: Connected Lighting Architecture

❖ Smart Parking

Parking is a universal challenge for cities around the globe. Ineffective parking access and administration make parking in urban areas a constant struggle and affect cities in many ways.

➔ **Smart Parking Use Cases:** Added traffic congestion is one consequence of drivers looking for parking space, and it has several consequences.

- Contributes to pollution
- Causes motorist frustration
- Increases traffic incidents
- Cities often lose revenue
- Parking administration employee productivity suffers
- Parking availability affects income

➔ **Smart Parking Architecture:** A variety of parking sensors are available on the market, and they take different approaches to sensing occupancy for parking spots. In high-density environments (for example, indoor parking, parking decks), one or several gateways per floor may connect to the parking sensors, using shorter-range protocols such as ZigBee or Wi-Fi. The gateway may then use another protocol (wired or wireless) to connect to the control station. In larger (for example, outdoor) environments, a longer-range Low Power Wide Area (LPWA) protocol is common, as shown in Figure

Smart+Connected Parking High-Level Architecture

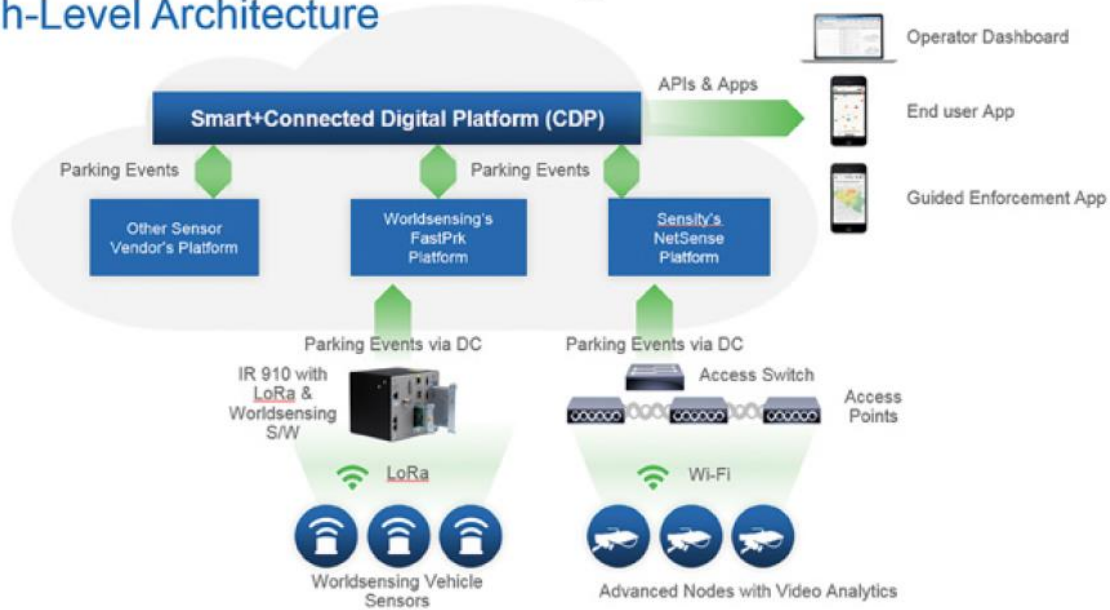


Figure: Connected Parking Architecture

Regardless of the technology used, 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.

Smart parking has three users that applications must support through aggregated data: city operators, parking enforcement personnel, and citizens. The following are some potential user experiences for these three user types.

- **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.
- **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.
- **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.

❖ 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

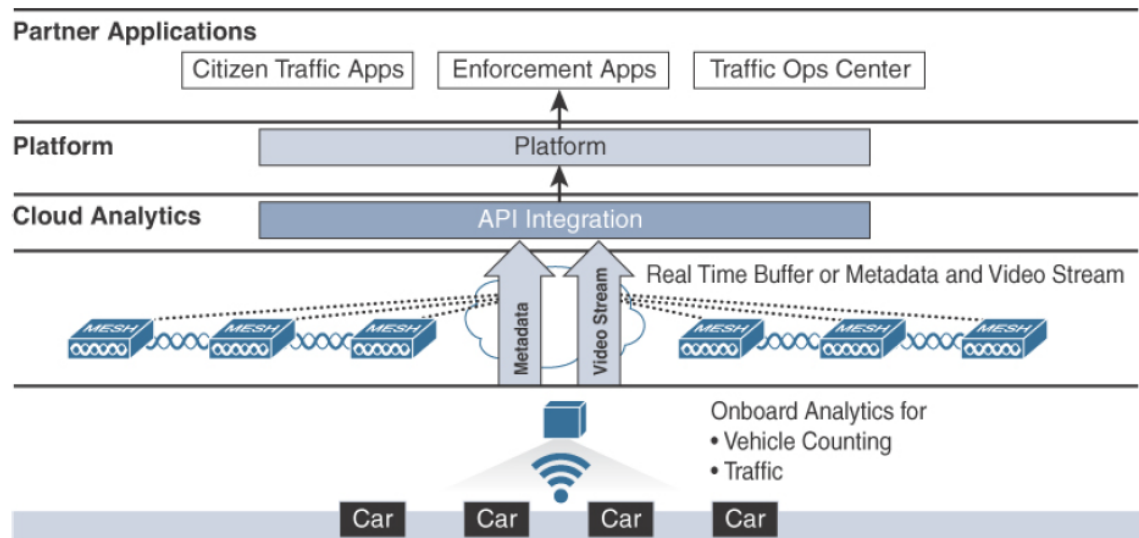


Figure: Smart city Traffic architecture

In the architecture shown in Figure, 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 co-ordination 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.

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. Communication techniques are as varied as sensor form factors.

❖ **Connected Environment**

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. 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:

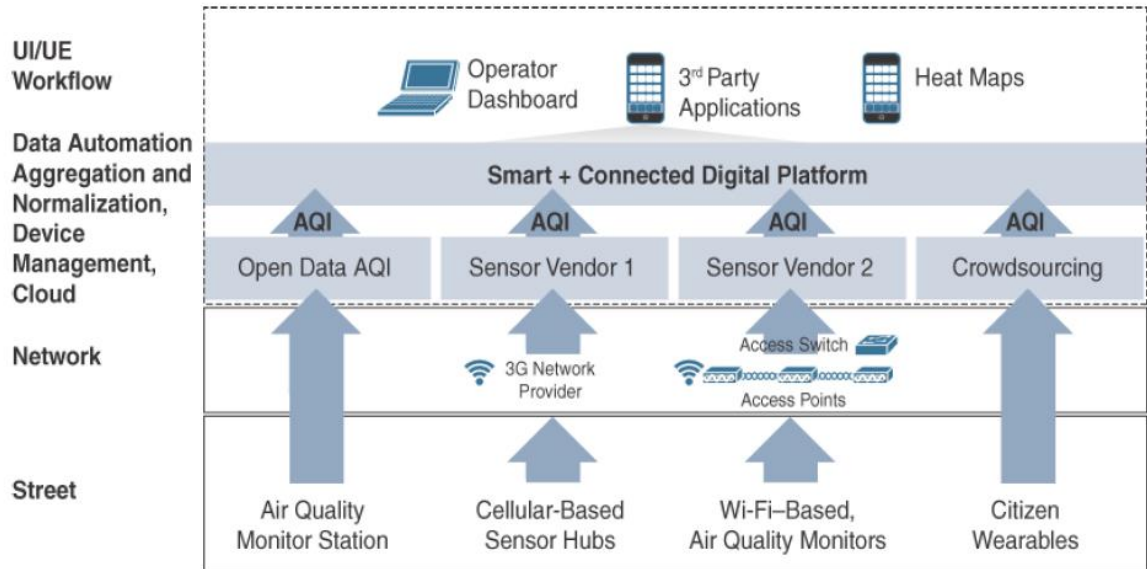
- Open-data platforms that provide current air quality measurements from existing air quality monitoring stations
- Sensors that provide similar accuracy to the air quality stations but are available at much lower prices
- Actionable insights and triggers to improve air quality through cross-domain actions
- Visualization of environmental data for consumers and maintenance of historical air quality data records to track emissions over time

Connected Environment Architecture

As shown in Figure, at the street layer there are a variety of multivendor sensor 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)



Independent and standalone sensors typically use wireless technologies. In dense urban environments, ZigBee and Wi-Fi are common. 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 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. This information can be visualized in applications that include heat maps of particulates, concentrates, and specific information on the dangers of such gaseous anomalies. Different pollution levels can be communicated, and gases can be tracked as they move throughout the city, either because of the wind or because of the movement of gas sources. From this pollution and environmental data and the analytics applied to it, the city can track problem areas and take action in long-term urban planning to reduce the effects of air quality disturbances.