CBCS SCHEME

USN | 1 | CR | 19 | IS | 036 | | | | | | 18CS81

Eighth Semester B.E. Degree Examination, June/July 2023
Internet of Things

Time: 3 hrs. Max. Marks: 100

Note: Answer any FIVE full questions, choosing ONE full question from each module.

Module-1

1 a. Define IOT. Explain in detail IOT and digitization. **(06 Marks)**
b. Explain in detail with any two example IOT impact. **(08 Marks)**
c. Explain the different evolutionary phases of the Internet. **(06 Marks)**

OR

2 a. Explain different challenges of IOT. **(04 Marks)**
b. Explain in detail IOT World Forum (IOTWF) Standard Architecture. **(08 Marks)**
c. Explain expanded view of the simplified IOT Architecture. **(08 Marks)**

Module-2

3 a. List and explain different types of sensors (any 8) with an example each. **(08 Marks)**
b. What are smart objects? With neat diagram, explain characteristics of smart object. **(08 Marks)**
c. What are Actuators? Explain comparison of sensors and actuators functionality with human. **(04 Marks)**

OR

4 a. What is SANET? Explain some advantages and disadvantages that a wireless based solution offers. **(06 Marks)**
b. Briefly explain protocol stack utilization IEEE 802.15.4. **(08 Marks)**
c. List and explain in brief communication criteria. **(06 Marks)**

Module-3

5 a. Explain key advantages of Internet Protocol for the IOT. **(08 Marks)**
b. Explain the need for optimization. **(08 Marks)**
c. With a neat diagram, explain comparison of an IOT protocol stack utilizing 6LOWPAN and IP protocol stack. **(04 Marks)**

OR

6 a. Write notes on Supervisory Control and Data Acquisition (SCADA). **(06 Marks)**
b. Explain with neat diagram Constrained Application Protocol (COAP) message format. **(08 Marks)**
c. Explain in detail Message Queuing Telemetry Transport (MQTT) publish/subscribe frame. **(06 Marks)**

Module-4

7 a. Compare: (i) Structured versus unstructured data (ii) Data in motion versus data at rest **(06 Marks)**
b. With neat diagram, explain Hadoop distributed cluster and writing a file to HDFS. **(08 Marks)**
c. Explain Lambda Architecture. **(06 Marks)**

OR

8 a. Explain edge streaming analytics and functions of Edge Analytics Processing Unit. **(10 Marks)**

b. Explain in detail formal risk analysis structures. **(10 Marks)**

Module-5

9 a. Write notes on: (i) Arduino UNO (ii) Raspberry Pi **(12 Marks)**

b. With a neat diagram, explain wireless temperature monitoring system using Raspberry Pi. **(08 Marks)**

OR

10 a. Explain IOT strategy for smarter cities. **(10 Marks)**

b. With neat smart cities Layered Architecture diagram, explain Smart City IOT Architecture. **(10 Marks)**

*****

VTU QP SOLUTION-JUNE-JULY-23

Subject Code – 18CS81 Subject – Internet of Things

1 A. DEFINE IOT. EXPLAIN IN DETAIL IOT AND DIGITIZATION

IoT is focused on connecting "things" to Internet.

Example: Wi-Fi location tracking in shopping mall

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

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

Example: Digitization of Photography

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

Example: Digitization of Taxi services

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

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

Example: "Nest" home automation

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

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

- The basic premise and goal of IoT is to "connect the unconnected."
- The world of IoT is broad and multifaceted.
- IoT is good to view it as an umbrella of various concepts, protocols, and technologies
- The age of IoT is often said to have started between the years 2008 and 2009.

VTU QP SOLUTION-JUNE-JULY-23

Subject Code – 18CS81 Subject – Internet of Things

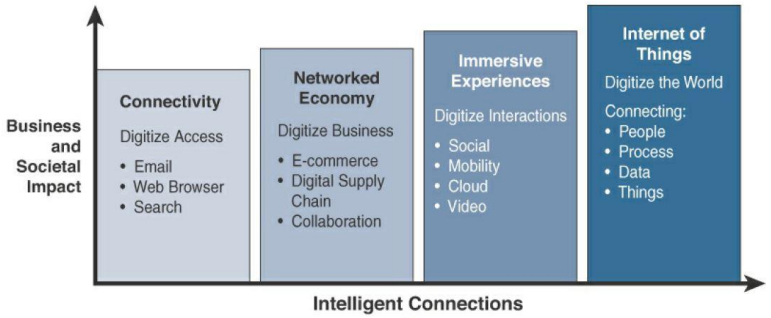


Figure 1-1 Evolutionary Phases of the Internet

1b. EXPLAIN IN DETAIL WITH ANY TWO EXAMPLE IOT IMPACT

Connected Roadways

IoT→

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

A self-driving car designed by Google operations:

- Basic sensors in cars monitor oil pressure, tire pressure, temperature, and other core car functions.
- The driver can access these data while controlling the car using equipment such as a steering wheel, pedals, and so on.
- The driver will understand, handle, and make critical decisions while concentrating on driving safely.
- IP-enabled sensors allow easy communication with other systems both inside and outside the car.
- Sensors and communication technologies vehicles to "talk" to other vehicles, traffic signals, school zones, and other elements of the transportation infrastructure.

Transportation challenges can be classified into the three categories:

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

Benefits of connected roadways :

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

IoT applications in connected roadways-

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

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

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

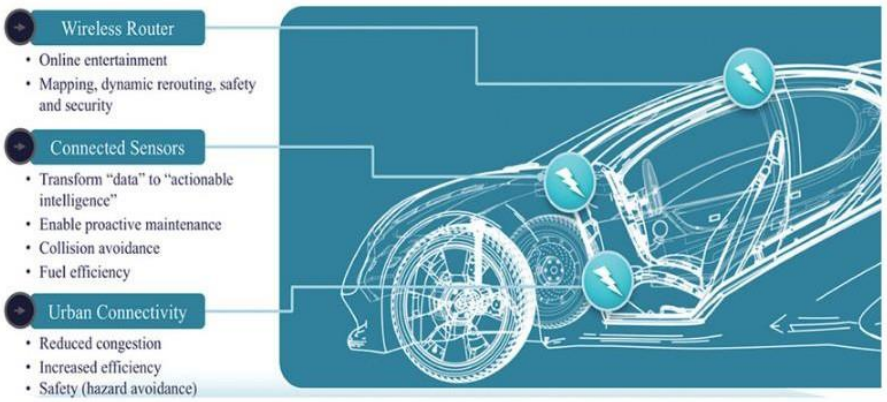


Figure 1-5 *The Connected Car*

- Tire companies can collect data related to use and durability of their products in real time.
❖ Automobile manufacturers collect to better understand how the cars are driven, when parts are starting to fail, or whether the car has broken down to build better cars in the future.

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

customized to optimize a road trip.

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

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

1 C EXPLAIN THE DIFFERENT EYALUTIONARY PHASES OF THE INTERNET

The main challenges facing manufacturing in a factory:

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

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

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

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

Example: real-time location system (RTLS).

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

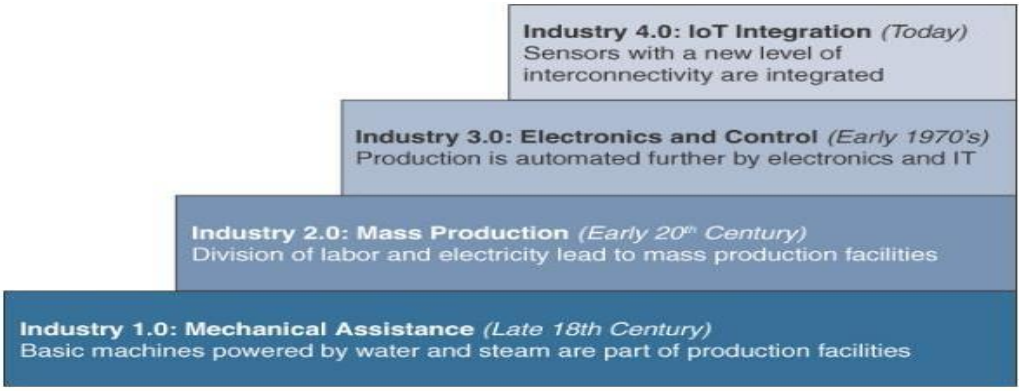


Figure 1-6 *The Four Industrial Revolutions*

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

2A. EXPLAIN DIFFERENT CHALLENGES OF IOT

- Scale
- Security
- Privacy
- Big data & Data Analytics
- Interoperability

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

Table 1-4 *IoT Challenges*

2B EXPLAIN IN DETAIL IÜT WORLD FORUM (IOTWF),STANDARD ARCHITECTURE

IoT World Forum (IoTWF) Standardized Architecture

The seven layers of the IoT Reference Model:

Layer 1: Physical Devices and Controllers Layer

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

Layer 2: Connectivity Layer

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

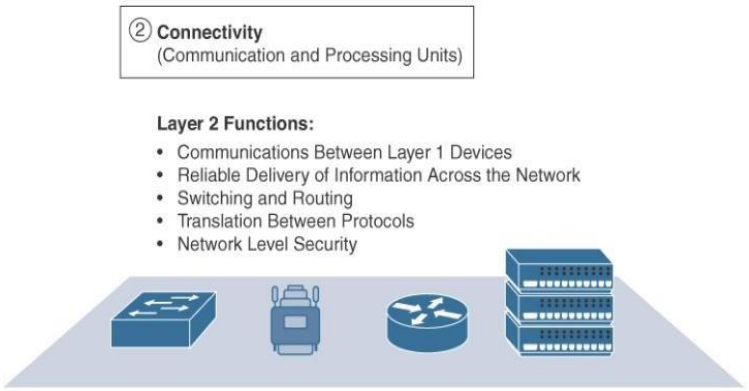


Figure 2-3 *IoT Reference Model Connectivity Layer Functions*

Layer 3: Edge Computing Layer

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

One of the basic principles of this reference model is that information processing is initiated as early and as close to the edge of the network as possible.

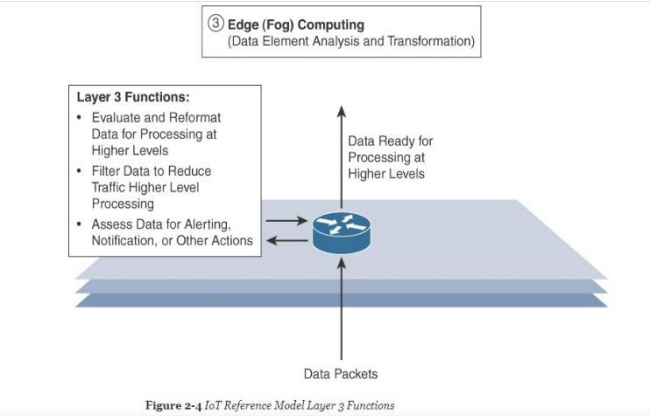


Figure 2-4 *IoT Reference Model Layer 3 Functions*

Upper Layers: Layers 4–7

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

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

IT and OT Responsibilities in the IoT Reference Model

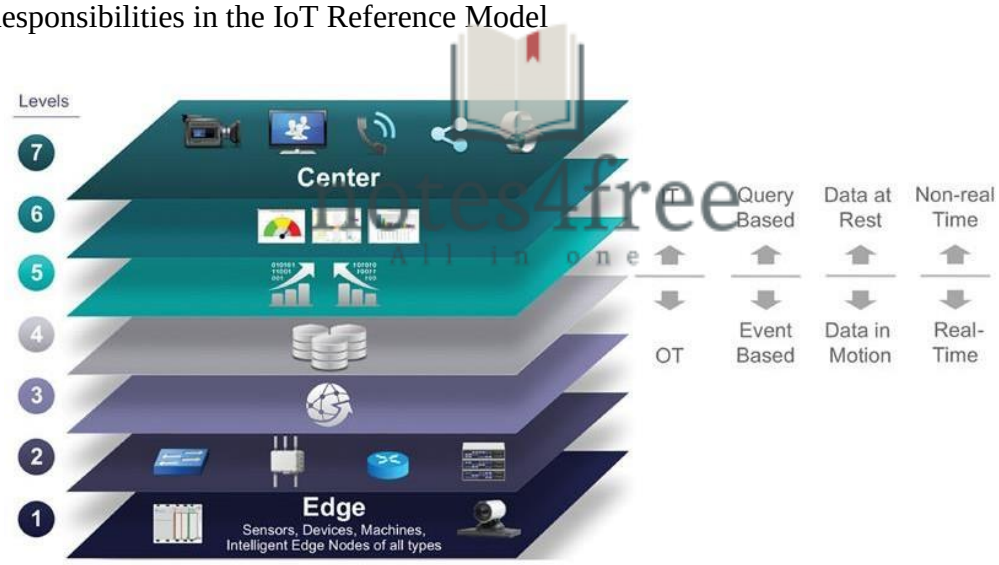


Figure 2-5 *IoT Reference Model Separation of IT and OT*

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

VTU QP SOLUTION-JUNE-JULY-23

Subject Code – 18CS81 Subject – Internet of Things

2C EXPLAIN EXPANDED VIEW OF THE SIMPLIFIED IoT ARCHITECTURE

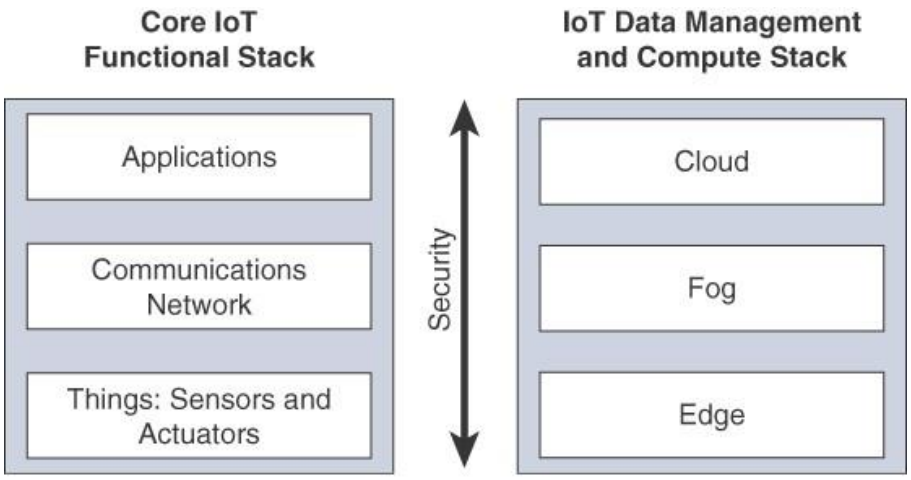


Figure 2-6 *Simplified IoT Architecture*

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

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

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

The three data management layers are the edge layer (data management within the sensors themselves),

the fog layer (data management in the gateways and transit network), and the cloud layer (data management in the cloud or central data center).

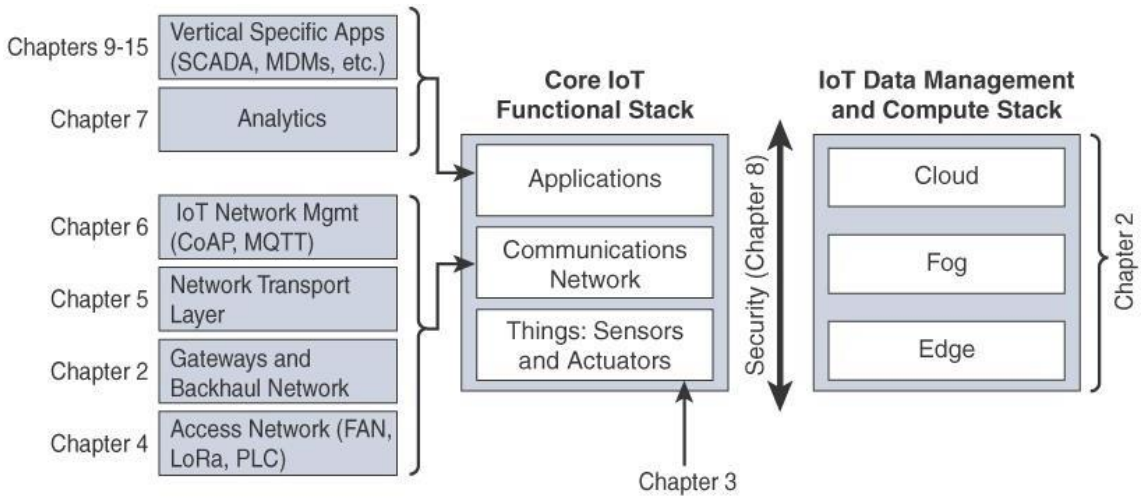


Figure 2-7 *Expanded View of the Simplified IoT Architecture*

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

3A LIST ANDEXPLAIN DIFFERENT TYPES OF SENSORS (ANY 8) WITH AN EXAMPLE EACH.

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

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

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

- **Active or passive:** Sensors can be categorized based on whether they produce an energy output and typically require an external power supply (active) or whether they simply receive energy and typically require no external power supply (passive).
- **Invasive or non-invasive:** Sensors can be categorized based on whether a sensor is part of the environment it is measuring (invasive) or external to it (non-invasive).

- **Contact or no-contact**: Sensors can be categorized based on whether they require physical contact with what they are measuring (contact) or not (no-contact).
- **Absolute or relative:** Sensors can be categorized based on whether they measure on an absolute scale (absolute) or based on a difference with a fixed or variable reference value (relative).
- **Area of application:** Sensors can be categorized based on the specific industry or vertical where they are being used.
- **How sensors measure:** Sensors can be categorized based on the physical mechanism used to measure sensory input (for example, thermoelectric, electrochemical, piezoresistive, optic, electric, fluid mechanic, photoelastic).
- **What sensors measure:** Sensors can be categorized based on their applications or what physical variables they measure.

VTU QP SOLUTION-JUNE-JULY-23

Subject Code – 18CS81 Subject – Internet of Things

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

3B WHAT ARE SMART OBJECTS? WITH NEAT DIAGRAM, EXPLAIN CHARACTERISTICS OF SMART OBJECT.

- ✓ Sensors come in all shapes and sizes and, can measure all types of physical conditions. A

fascinating use case to highlight the power of sensors and IoT is in the area ofprecision agriculture (smart farming), which uses a variety of technical advances to improve the efficiency, sustainability, and profitability of traditional farming practices.

- ✓ This includes the use of GPS and satellite aerial imagery for determining field viability; robots for high-precision planting, harvesting, irrigation, and so on; and real-time analytics and artificial intelligence to predict optimal crop yield, weather impacts, and soil quality.
- ✓ The astounding volume of sensors is in large part due to their smaller size, their form factor, andtheir decreasing cost. These factors make possible the economic and technical feasibility of havingan increased density of sensors in objects of all types.
- ✓ Perhaps the most significant accelerator for sensor deployments is mobile phones. More than a billion smart phones are sold each year, and each one has well over a dozen sensors inside it (see Figure 3-2), and that number continues to grow each year.

Figure 3-2 *Sensors in a Smart Phone*

VTU QP SOLUTION-JUNE-JULY-23

Subject Code – 18CS81 Subject – Internet of Things

3C WHAT ARE ACTUATORS? EXPLAN COMPARISON OF SENSORS AND ACTUATORS FUNCTIONALITY WITH HUMAN.

The following are some advantages 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

✓ Actuators are natural complements to sensors.

✓ Actuators receive some type of control signal (commonly an electric signal or digital command) that triggers a physical effect, usually some type of motion, force, and so on.

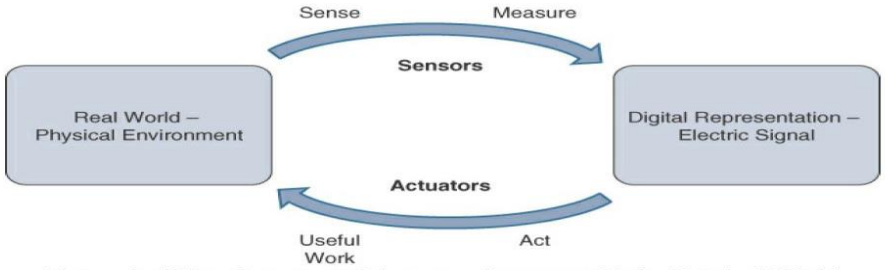


Figure 3-4 How Sensors and Actuators Interact with the Physical World

4A WHAT IS SANETEXPLAIN SOME ADVANTAGES ÄND DISADVANTAGES THAT A WIRELESS BASED SOLUTION OFFERS.

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

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

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

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

Advantages:

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

Disadvantages:

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

Wireless Sensor Networks (WSNs)

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

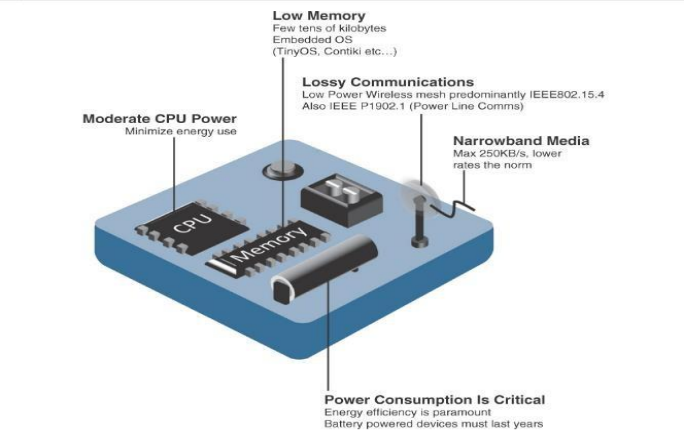


Figure 3-8 *Design Constraints for Wireless Smart Objects*

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

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

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

4B. BRIEFLY EXPLAIN PROTOCOL STACK UTILIZÁTION 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.
- 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

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.

Physical Layer:

- The 802.15.4 standard supports an extensive number of PHY options that range from 2.4 GHz to sub-GHz frequencies in ISM bands.
- original IEEE 802.15.4-2003 standard specified only 3 PHY options based on **Direct Sequence Spread Spectrum (DSSS)** modulation.

The original physical layer transmission options were as follows:

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

MAC Layer: 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.

Topology:

- IEEE 802.15.4–based networks can be built as star, peer-to-peer, or mesh topologies.
- **Mesh networks tie together many nodes.**
- A minimum of 1 **FFD** acting as a PAN coordinator **is required** to deliver services that allow other devices to form a cell or PAN.

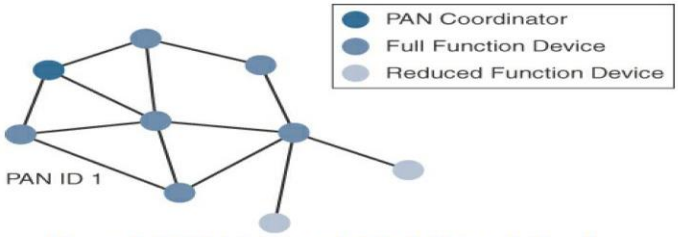


Figure 4-7 802.15.4 Sample Mesh Network Topology

Security:

- The IEEE 802.15.4 specification uses Advanced Encryption Standard (AES) with a 128-bit key length.
- In addition to encrypting the data, AES here, also validates the data.
- Enabling these security features for 802.15.4 changes the frame format slightly.

Competitive Technologies:

- A competitive radio technology that is different in its PHY and MAC layers is DASH7.
- DASH7 was used by US military forces for many years, mainly for logistics purposes.
- The current DASH7 technology offers low power consumption, a compact protocol stack, range up to 1 mile, and AES encryption.
- DASH7 is promoted by the DASH7 Alliance.

4C LIST AND EXPLAIN IN BRIEF COMMUNICATION CRITERIA

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

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

3.5.1.1 Range

VTU QP SOLUTION-JUNE-JULY-23

Subject Code – 18CS81 Subject – Internet of Things

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

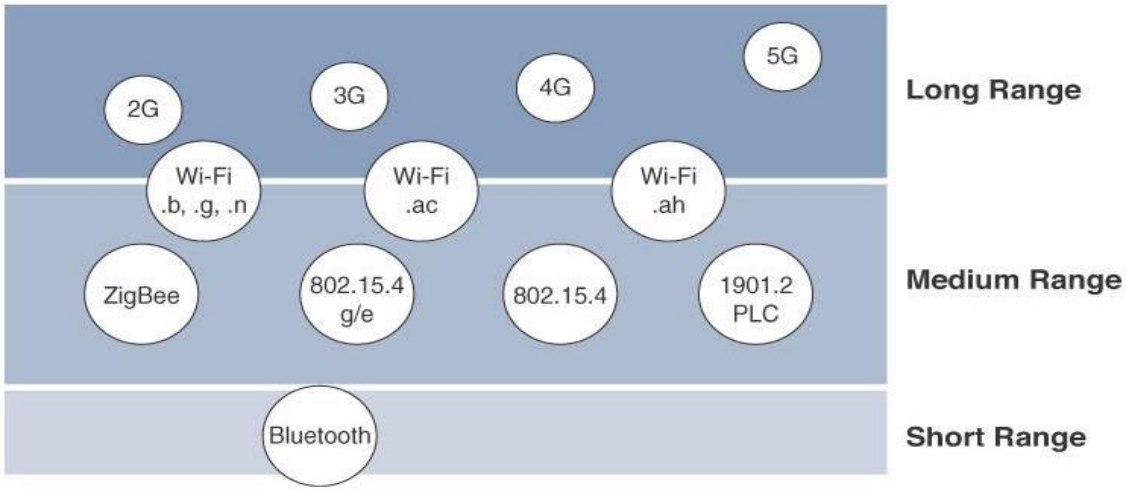


Figure 4-1 *Wireless Access Landscape*

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

5A EXPLAIN KEY ADVANTAGES,OT INTEMET PROTOCOL FOR THE IOT.

One of the main differences between traditional information technology (IT) and operational technology (OT)is the lifetime of the underlying technologies and products. One way to guarantee multi-year lifetimes is todefine a layered architecture such as the 30-year-old IP architecture.

IP has largely demonstrated its ability tointegrate small and large evolutions. At the same time, it is able to maintain its operations for large numbers of devices and users, such as the 3 billion Internet users. Before evaluating the pros and cons of IP adoption versus adaptation, this section provides a quick review of the key advantages of the IP suite for the Internet ofThings:

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

- **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. Even if physical and data link layers such as Ethernet, Wi-Fi, and cellular are widely adopted, the history of data communications demonstrates that no given wired or wireless technology fits all deployment criteria. Furthermore, communication technologies evolve at a pace faster than the expected 10- to 20- year lifetime of OT solutions. So, the layered IP architecture is well equipped to cope with any type of physical and data link layers. This makes IP ideal as a long-term investment because various protocols at these layers can be used in a deployment now and over time, without requiring changes to the whole solution architecture and data flow.

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

- **Scalable:** As the common protocol of the Internet, IP has been massively deployed and tested for robust scalability. Millions of private and public IP infrastructure nodes have been operational for years, offering strong foundations for those not familiar with IP network management. Of course, adding huge numbers of "things" to private and public infrastructures may require optimizations and design rules specific to the new devices.
- **Manageable and highly secure:** Communications infrastructure requires appropriate management and security capabilities for proper operations. One of the benefits that comes from 30 years of operational IP networks is the well-understood network management and security protocols, mechanisms, and toolsets that are widely available. Adopting IP network management also brings an operational business application to OT. Well-known network and security management tools are easily leveraged with an IP network layer.
- **Stable and resilient:** IP has been around for 30 years, and it is clear that IP is a workable solution. IP has a large and well-established knowledge base and, more importantly, it has been used for years in critical infrastructures, such as financial and defense networks. In addition, IP has been deployed for critical services, such as voice and video, which have already transitioned from closed environments to open IP standards. Finally, its stability and resiliency benefit from the large ecosystem of IT professionals who can help design, deploy, and operate IP-based solutions.
- **Consumers' market adoption:** When developing IoT solutions and products targeting the consumer market, vendors know that consumers' access to applications and devices will occur predominantly over broadband and mobile wireless infrastructure. The main consumer devices range from smart phones to tablets and PCs. The common protocol that links IoT in the consumer space to these devices is IP.
- **The innovation factor:** The past two decades have largely established the adoption of IP as a factor for increased innovation. IP is the underlying protocol for applications ranging from file transfer and e-mail to the World Wide Web, e-commerce, social networking, mobility, and more. Even the recent computing evolution from PC to mobile and mainframes to cloud services are perfect demonstrations of the innovative ground enabled by IP.

5.B. EXPLAIN THE NEED FOR OPTIMIZATION.

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

Constrained Nodes

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

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

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

While it has been largely demonstrated that production IP stacks perform well in constrained nodes, classification of these nodes helps when evaluating the IP adoption versus adaptation model selection. IoT constrained nodes can be classified as follows:

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

Constrained Networks

Constrained networks have unique characteristics and requirements. In contrast with typical IP networks, where highly stable and fast links are available, constrained networks are limited by low- power, low-bandwidth links (wireless and wired). They operate between a few kbps and a few hundred kbps and may utilize a star, mesh, or combined network topologies, ensuring proper operations. With a constrained network,

in addition to limited bandwidth, itis not unusual for the packet delivery rate (PDR) to oscillate between low and high percentages. Large bursts of unpredictable errors and even loss of connectivity at times may occur. These behaviors can be observed on both wireless and narrowband power-line communication links, where packet delivery variation may fluctuate greatly during the course of a day.

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

VTU QP SOLUTION-JUNE-JULY-23

Subject Code – 18CS81 Subject – Internet of Things

IP Versions

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

While it may seem natural to base all IoT deployments on IPv6, current infrastructures and their associated lifecycle of solutions, protocols, and products need to be taken into account. IPv4 is entrenched in these current infrastructures, and so support for it is required in most cases. Therefore, the Internet of Things has to follow a similar path as the Internet itself and support both IPv4 and IPv6 versions concurrently. Techniques such as tunneling and translation need to be employed in IoT solutions to ensure interoperability between IPv4 and IPv6.

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

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

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

(Narrowband Power Line Communications) only have an IPv6 adaptation layer specified. This means that any device implementing a technology that requires an IPv6 adaptation layer must communicate over an IPv6-only subnetwork. This is reinforced by the IETF routing protocol for LLNs, RPL, which is IPv6 only.

5 C WITH A NEAT DIAGRAM, EXPLAIN COMPARISON OF IOT PROTOCOL STACK UTILIZING 6LOWPAN AND IP PROTOCOL STACK.

- ✓ In the IP architecture, the transport of IP packets over any given Layer 1 (PHY) and Layer 2 (MAC) protocolmust be defined and documented.
- ✓ The model for packaging IP into lower-layer protocols is often referred to as an *adaptation layer*.
- ✓ Unless the technology is proprietary, IP adaptation layers are typically defined by an IETF working groupand released as a Request for Comments (RFC).
- ✓ An RFC is a publication from the IETF that officially documents Internet standards, specifications, protocols, procedures, and events. For example, RFC 864 describes how an IPv4 packet gets encapsulated over an Ethernet frame, and RFC 2464 describes how thesame function is performed for an IPv6 packet.
- ✓ IoT-related protocols follow a similar process. The main difference is that an adaptation layer designed for
- ✓ IoT may include some optimizations to deal with constrained nodes and networks.
- ✓ The main examples of adaptation layers optimized for constrained nodes or "things" are the ones under the 6LoWPAN working groupand 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 shows an example of an IoT protocol stack using the 6LoWPAN adaptation layer beside the well- known IP protocol stack for reference.

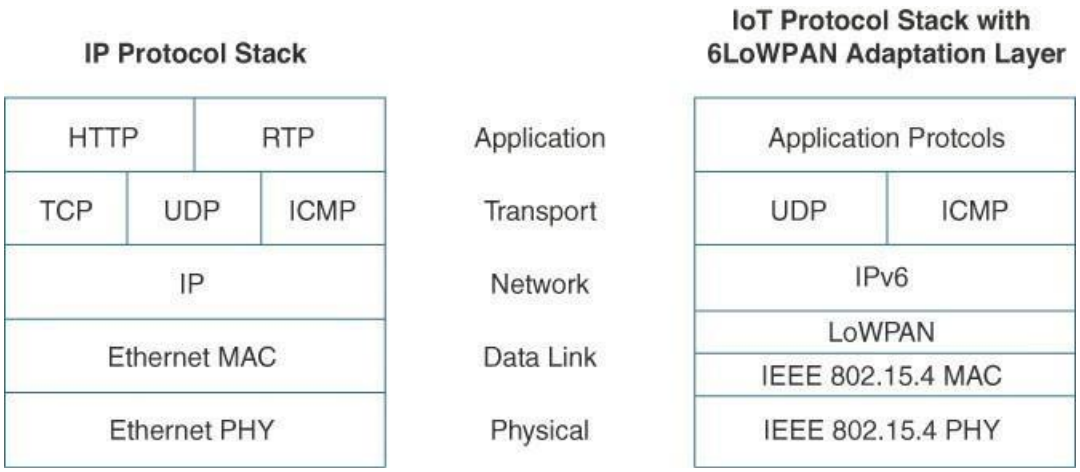


Figure 3.2 *Comparison of an IoT Protocol Stack Utilizing 6LoWPAN and an IP Protocol Stack*

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

Header Compression

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

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

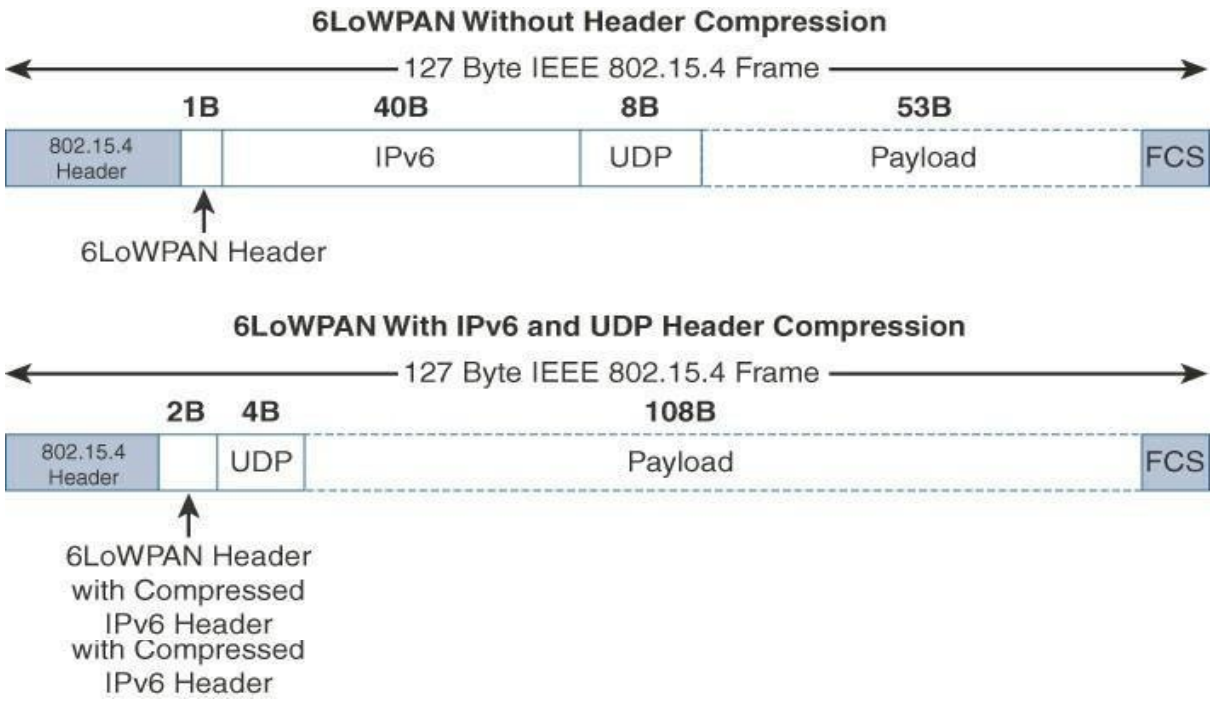


Figure 3.4 *6LoWPAN Header Compression*

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

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

3.3.1.1 Fragmentation

The maximum transmission unit (MTU) for an IPv6 network must be at least 1280 bytes. The term *MTU* defines the size of the largest protocol data unit that can be passed. For IEEE 802.15.4, 127 bytes is the MTU. Because of this is there is a problem that IPv6 with a much larger MTU, is carried inside the 802.15.4 frame with a much smaller one. To remedy this situation, large IPv6 packets must be fragmented across multiple 802.15.4 frames at Layer 2.

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

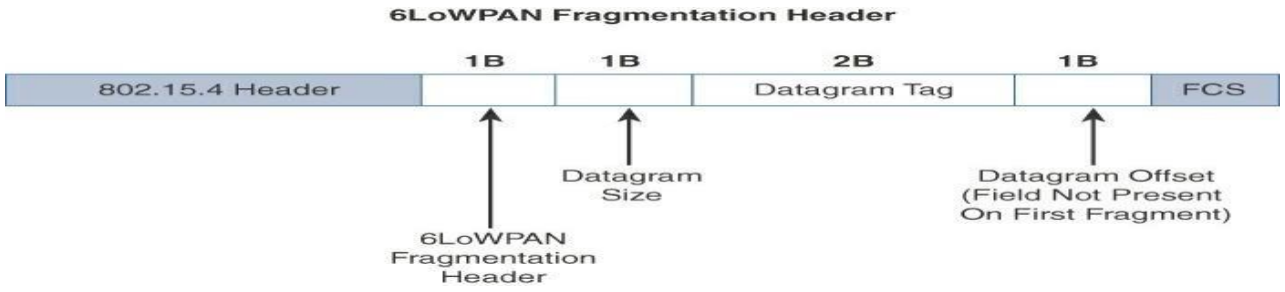


Figure 3.5 *6LoWPAN Fragmentation Header*

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

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

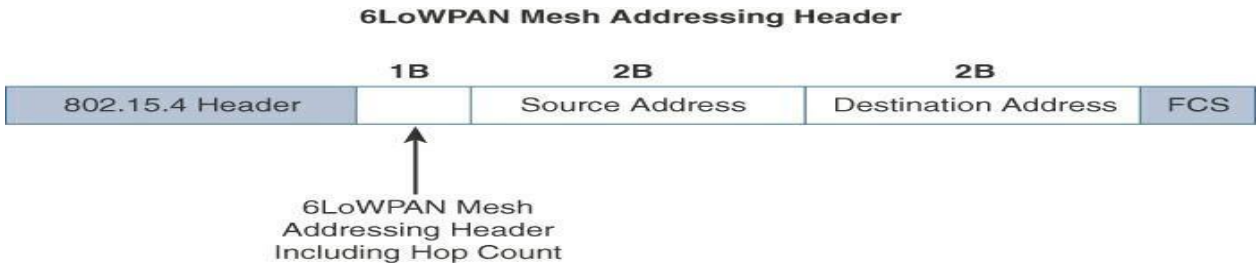


Figure 3.6 *6LoWPAN Mesh Addressing Header*

Mesh-Under Versus Mesh-Over Routing

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

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

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

MMMMMWorking Group

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

Therefore, the charter of the 6Lo working group, now called the IPv6 over Networks of Resource- Constrained Nodes, is to facilitate the IPv6 connectivity over constrained-node networks. In particular, this working group

is focused on the following:

- **IPv6-over-foo adaptation layer specifications using 6LoWPAN technologies (RFC4944, RFC6282, RFC6775) for link layer technologies:** For example, this includes:
    - IPv6 over Bluetooth Low Energy
    - Transmission of IPv6 packets over near-field communication IPv6 over 802.11ah
    - Transmission of IPv6 packets over DECT Ultra Low Energy
    - Transmission of IPv6 packets on WIA-PA (Wireless Networks for Industrial Automation– Process Automation)
    - Transmission of IPv6 over Master Slave/Token Passing (MS/TP)
- **Information and data models such as MIB modules:** One example is RFC 7388, "Definition of Managed Objects for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)."
- **Optimizations that are applicable to more than one adaptation layer specification:** For example, this includes RFC 7400, "6LoWPAN-GHC: Generic Header Compression for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)."

Informational and maintenance publications needed for the IETF specifications in this area

6A WRITE NOTES ON SUPERVISORY CONTRÖB AND DATA ACQUISITION (SCADA).

Supervisory Control and Data Acquisition (SCADA)

- Its Automation control system, implemented without IP over serial links.
- SCADA systems collect sensor data and telemetry from remote devices, while also providing the ability to control them.
- SCADA networks can be found across various industries.

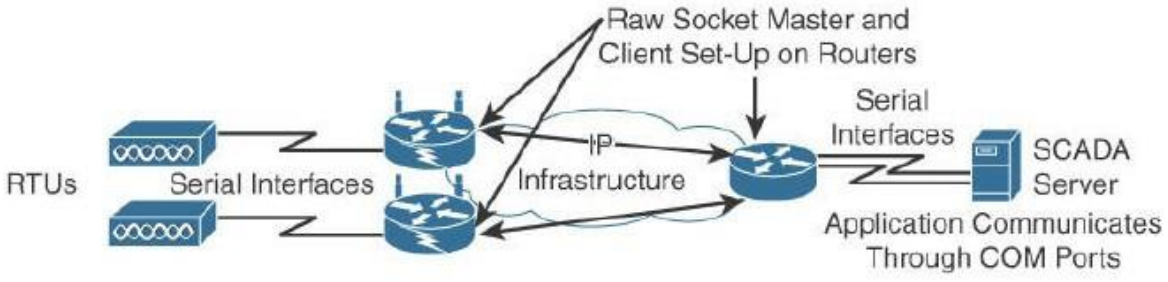
SCADA commonly uses certain protocols for communications between devices and applications

- Transport of the original serial protocol over IP can be achieved either by tunneling using raw sockets over TCP or UDP or by installing an intermediate device.
- Intermediate device may perform protocol translation between the serial protocol version and its IP implementation.
- A raw socket connection simply denotes that the serial data is being packaged directly into a TCP or UDP transport.
- A socket in this instance is a standard application programming interface (API) composed of an IP address and a TCP or UDP port that is used to access network devices over an IP network.
- Figures next detail raw socket scenarios for a legacy SCADA server trying to communicate with remote serial devices.
- Both the SCADA server and the remote terminal units (RTUs) have a direct serial connection to their respective routers.
- The routers terminate the serial connections at both ends of the link and use raw socket encapsulation to transport the serial payload over the IP network.

VTU QP SOLUTION-JUNE-JULY-23

Subject Code – 18CS81 Subject – Internet of Things

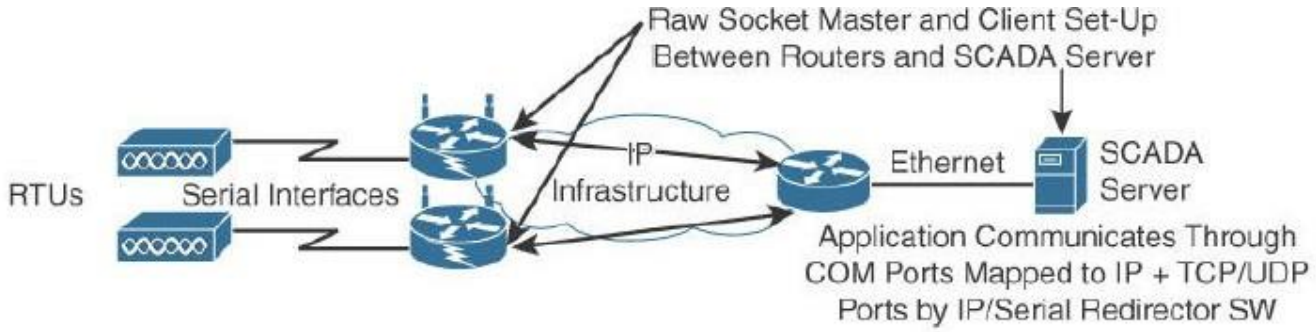
NOTE: An RTU is a multipurpose device used to monitor and control various systems, applications, and devices managing automation



Scenario A: Raw Socket between Routers – no change on SCADA server

- **A piece of software is installed on the SCADA server that maps the serial COM ports to IP ports.**
- **This software is commonly referred to as an IP/serial redirector.**

The IP/serial redirector in essence terminates the serial connection of the SCADA server and converts it to a TCP/IP port using a raw socket connection



Scenario B: Raw Socket between Router and SCADA Server – no SCADA application change on server but IP/Serial Redirector software and Ethernet interface to be added

6B. EXPLAIN WITH NEAT DIAGRAM CONSTRAINED APPLICATION PROTOCOL (COAP) MESSAGE FORMAT

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

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

including the following:

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

The CoAP messaging model is primarily designed to facilitate the exchange of messages over UDP between endpoints, including the secure transport protocol Datagram Transport Layer Security (DTLS). The IETF CoRE working group is studying alternate transport mechanisms, including TCP, secure TLS, and WebSocket. CoAP over Short Message Service (SMS) as defined in Open Mobile Alliance for Lightweight Machine-to-Machine (LWM2M) for IoT device management is also being considered.

RFC 7252 provides more details on securing CoAP with DTLS. It specifies how a CoAP endpoint is provisioned with keys and a filtering list. Four security modes are defined: NoSec, PreSharedKey, RawPublicKey, and Certificate. The NoSec and RawPublicKey implementations are mandatory.

From a formatting perspective, a CoAP message is composed of a short fixed-length Header field (4 bytes), a variable-length but mandatory Token field (0–8 bytes), Options fields if necessary, and the Payload field. Figure 3.17 details the CoAP message format, which delivers low overhead while decreasing parsing complexity.

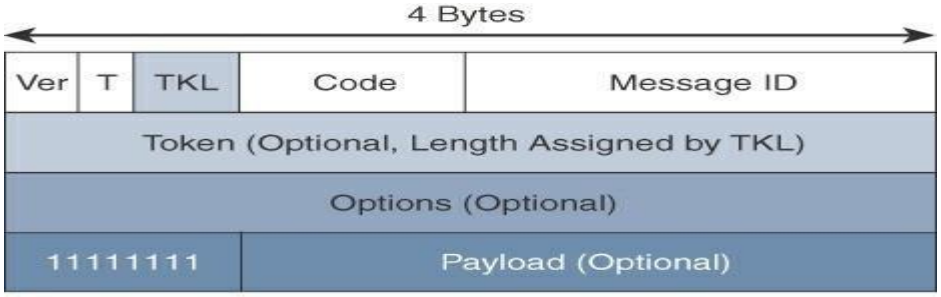


Figure 3.17 *CoAP Message Format*

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

Table 6-1 *CoAP Message Fields*

CoAP Message Field	Description
Ver (Version)	Identifies the CoAP version.
T (Type)	Defines one of the following four message types: Confirmable (CON), Non-confirmable (NON), Acknowledgement (ACK), or Reset (RST). CON and ACK are highlighted in more detail in Figure 6-9.
TKL (Token Length)	Specifies the size (0–8 Bytes) of the Token field.
Code	Indicates the request method for a request message and a response code for a response message. For example, in Figure 6-9, GET is the request method, and 2.05 is the response code. For a complete list

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

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

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

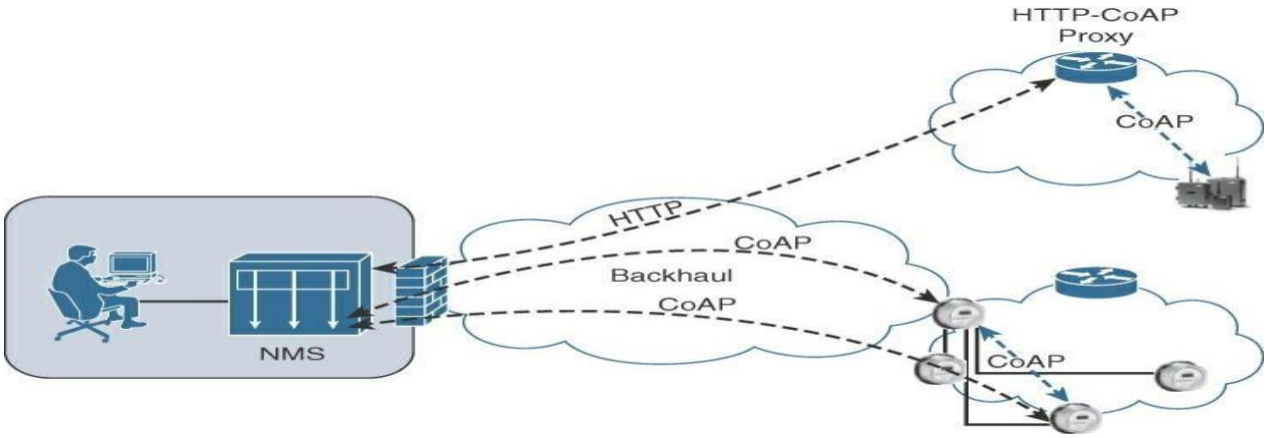


Figure 3.18 *CoAP Communications in IoT Infrastructures*

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

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

While running over UDP, CoAP offers a reliable transmission of messages when a CoAP header is marked as "confirmable." In addition, CoAP supports basic congestion control with a default time-out, simple stop and wait retransmission with exponential back-off mechanism, and detection of duplicate messages through a message ID. If a request or response is tagged as confirmable, the recipient must explicitly either acknowledge or reject the message, using the same message ID, as shown in Figure 3.19. If a recipient can't

process a non-confirmable message, a reset message is sent.

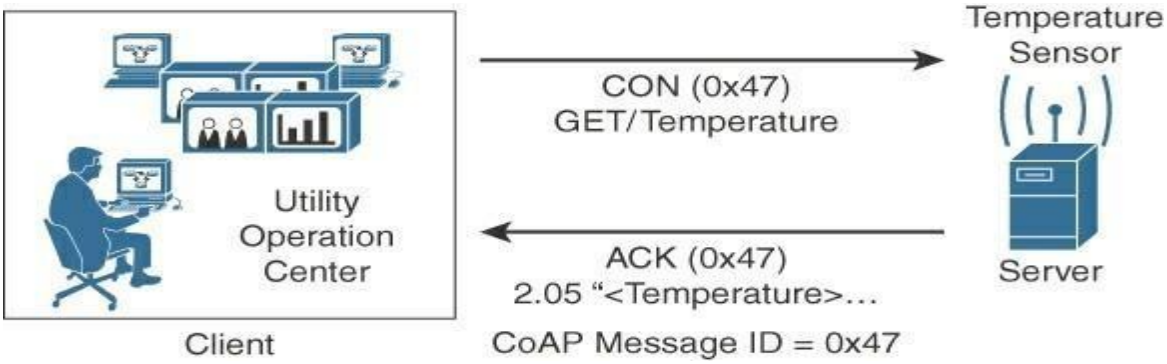


Figure 3.19 *CoAP Reliable Transmission Example*

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

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

With often no affordable manual configuration on the IoT endpoints, a CoAP server offering services and resources needs to be discovered by the CoAP clients. Services from a CoAP server can either be discovered by learning a URI in a namespace or through the "All CoAP nodes" multicast address. When utilizing the URI scheme for discovering services, the default port 5683 is used for non-secured CoAP, or **coap**, while port 5684 is utilized for DTLS-secured CoAP, or **coaps**. The CoAP server must be in listening state on these ports, unless a different port number is associated with the URI in a namespace. Much as with accessing web server resources, CoAP specifications provide a description of the relationships between resources in RFC 6690, "Constrained RESTful Environments (CoRE) Link Format."

To improve the response time and reduce bandwidth consumption, CoAP supports caching capabilities based on the response code. To use a cache entry, a CoAP endpoint must validate the presented request and stored response matches, including all options (unless marked as NoCacheKey). This confirms

that the stored response is fresh or valid. A wide range of CoAP implementations are available. Some are published with open source licenses, and others are part of vendor solutions

6C. EXPLAIN IN DETAIL MESSAGE QUEUING TELEMETRY TRANSPORT (MQTT) PUBLISH/SUBSCRIBE FRAME.

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

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

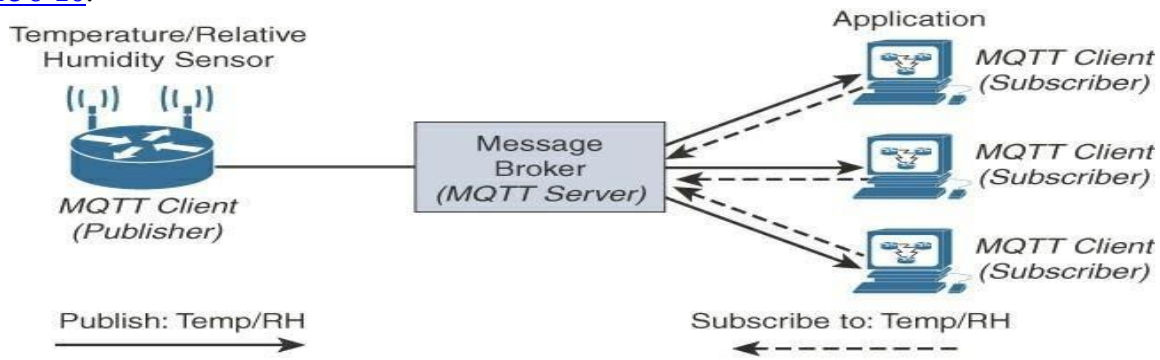


Figure 3.20 *MQTT Publish/Subscribe Framework*

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

The application on the right side of Figure 3.20 is an MQTT client that is a subscriber to the Temp/RH data being generated by the publisher or sensor on the left. This model, where subscribers express a desire to receive information from publishers, is well known. A great example is the collaboration and social networking application Twitter.

With MQTT, clients can subscribe to all data (using a wildcard character) or specific data from the information tree of a publisher. In addition, the presence of a message broker in MQTT decouples the data

transmission between clients acting as publishers and subscribers. In fact, publishers and subscribers do not even know (or need to know) about each other. A benefit of having this decoupling is that the MQTT message broker ensures that information can be buffered and cached in case of network failures. This also means that publishers and subscribers do not have to be online at the same time.

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

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

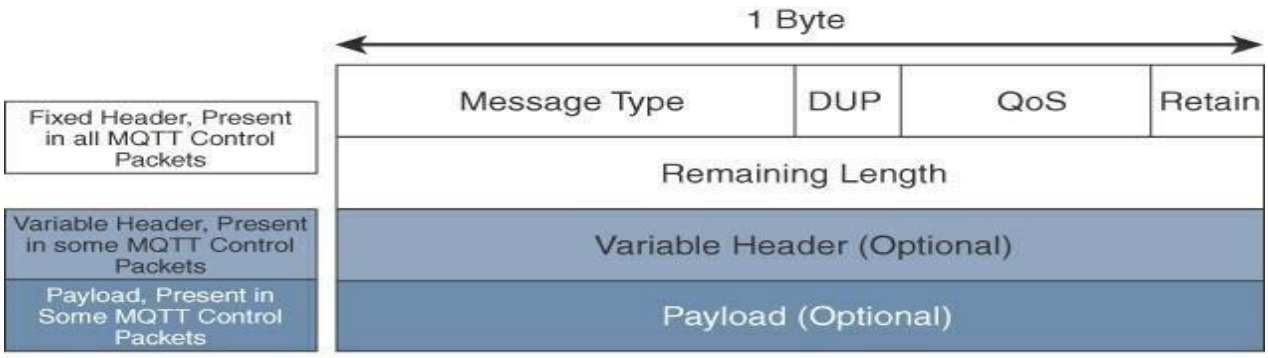


Figure 3.21 *MQTT Message Format*

Compared to the CoAP message format in Figure 3.17, MQTT contains a smaller header of 2 bytes compared to 4 bytes for CoAP. The first MQTT field in the header is Message Type, which identifies the kind of MQTT packet within a message. Fourteen different types of control packets are specified in MQTT version 3.1.1. Each of them has a unique value that is coded into the Message Type field. Note that values 0 and 15 are reserved. MQTT message types are summarized in Table 3.2.

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

Table 6-2 *MQTT Message Types*

VTU QP SOLUTION-JUNE-JULY-23

Subject Code – 18CS81 Subject – Internet of Things

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

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, which allows the identification of the MQTT session between both parties. When the server is delivering an application message to more than one client, each client is treated independently.

The MQTT protocol offers three levels of quality of service (QoS). QoS for MQTT is implemented when exchanging application messages with publishers or subscribers, and it is different fromthe IP QoS that most people are familiar with. The delivery protocol is symmetric. This means the client and server can each take the role of either sender or receiver. The delivery protocol is concerned solely with the delivery of an application message from a single sender to a single receiver. These are the three levels of MQTT 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. There is an increased overhead associated with this QoS level because each packet contains an optional variable header with a packet identifier. Confirming the receipt of a PUBLISH message requires a two-step acknowledgement process. The first step is done through the

PUBLISH/PUBREC packet pair, and the second is achieved with the PUBREL/PUBCOMP packet pair. This level provides a "guaranteed service" known as "exactly once" delivery, with no consideration for the number of retries as long as the message is delivered once.

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

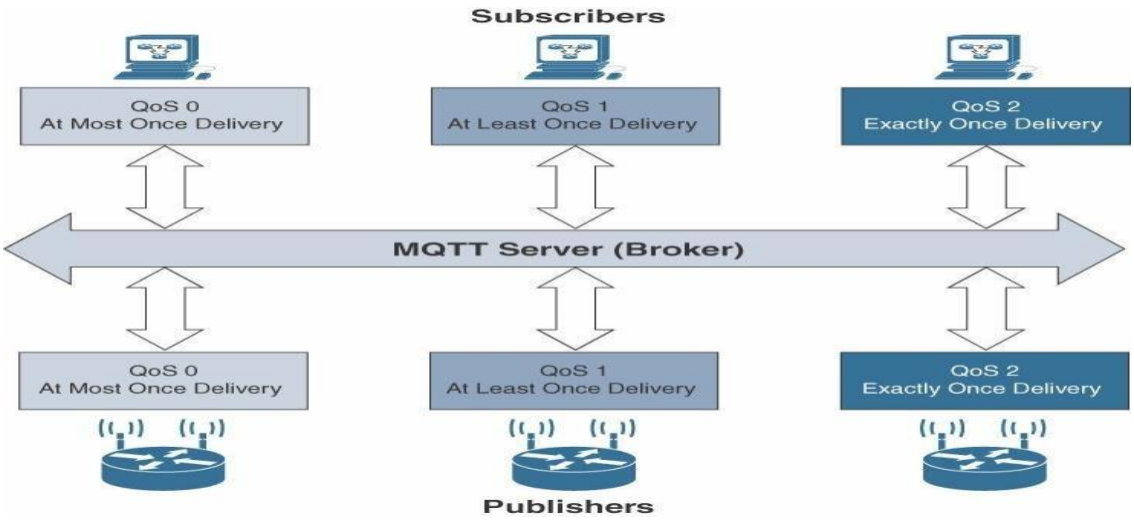


Figure 3.22 *MQTT QoS Flows*

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

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

7A. COMPARE: (I) STRUCTÜRED VERSUS UNSTRUCTURED DATA (I) DATA IN MOTION VERSUS DATA AT REST

7.1.1 Structured Versus Unstructured Data

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

structured data and unstructured data.

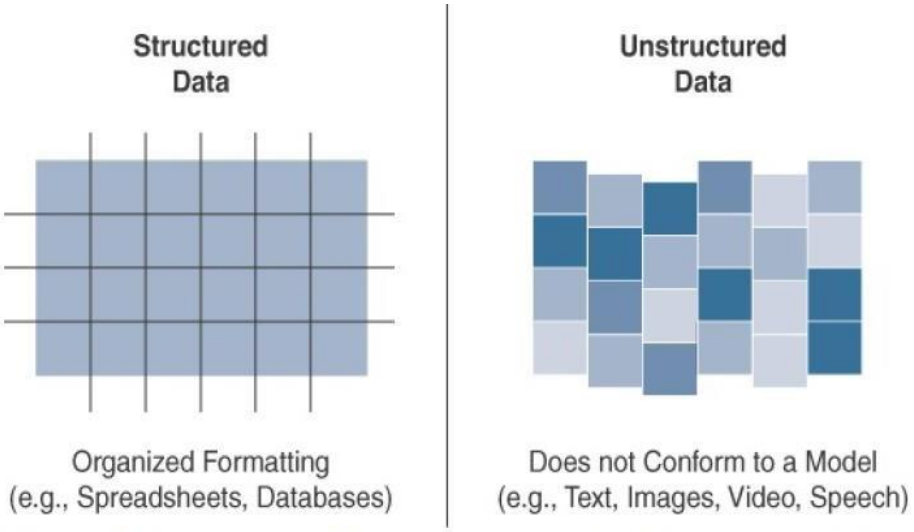


Figure 7-2 *Comparison Between Structured and Unstructured Data*

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

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

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

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

image/facial recognition applications, you can extract critical information from still images and video.

Data in Motion Versus Data at Rest

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

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

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

7B. WITH NEAT DIAGRAM, EXPLAIN HADOOP DISTRIBUTED CLUSTER AND WRITING A FILE TO HDES

Hadoop

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

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

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

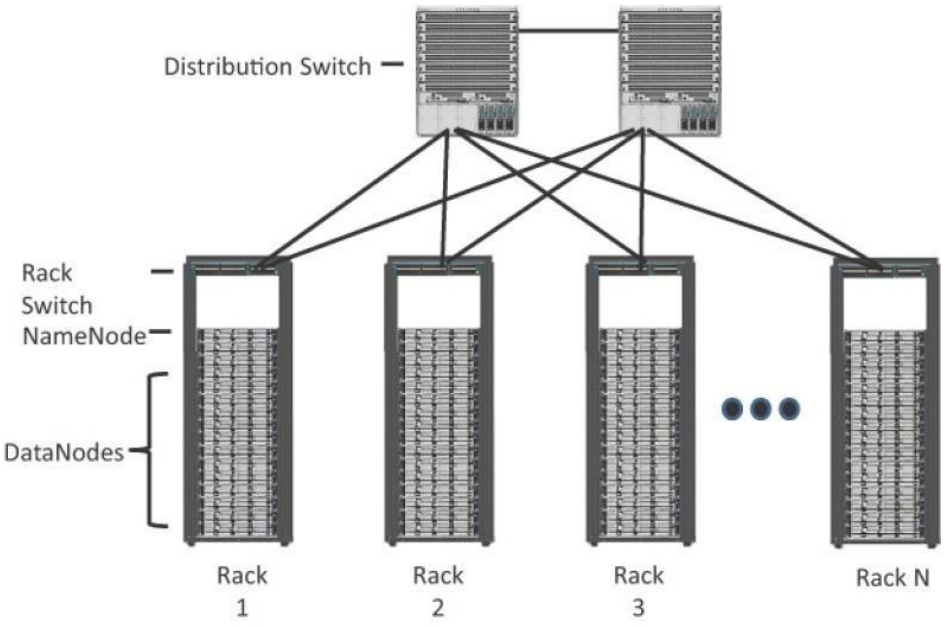


Figure 7-8 *Distributed Hadoop Cluster*

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

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

- **NameNodes:** These are a critical piece in data adds, moves, deletes, and reads on HDFS. They coordinate where the data is stored, and maintain a map of where each block of data is stored and where it is replicated. All interaction with HDFS is coordinated through the primary (active) NameNode, with a secondary (standby) NameNode notified of thechanges in the event of a failure of the primary. The NameNode takes write requests fromclients and distributes those files across the available nodes in configurable block sizes, usually 64 MB or 128 MB blocks. The NameNode is also responsible for instructing the DataNodes where replication should occur.
- **DataNodes**: These are the servers where the data is stored at the direction of the NameNode. It is common to have many DataNodes in a Hadoop cluster to store the data. Data blocks are distributed across several nodes and often are replicated three, four, or more times across nodes for redundancy. Once data is written to one of the DataNodes, the DataNode selects two (or more) additional nodes, based on replication policies, to ensure

data redundancy across the cluster. Disk redundancy techniques such as Redundant Array of Independent Disks (RAID) are generally not used for HDFS because the NameNodes and DataNodes coordinate blocklevel redundancy with this replication technique. Figure 7-9 shows the relationship between NameNodes and DataNodes and how data blocks are distributed across the cluster.

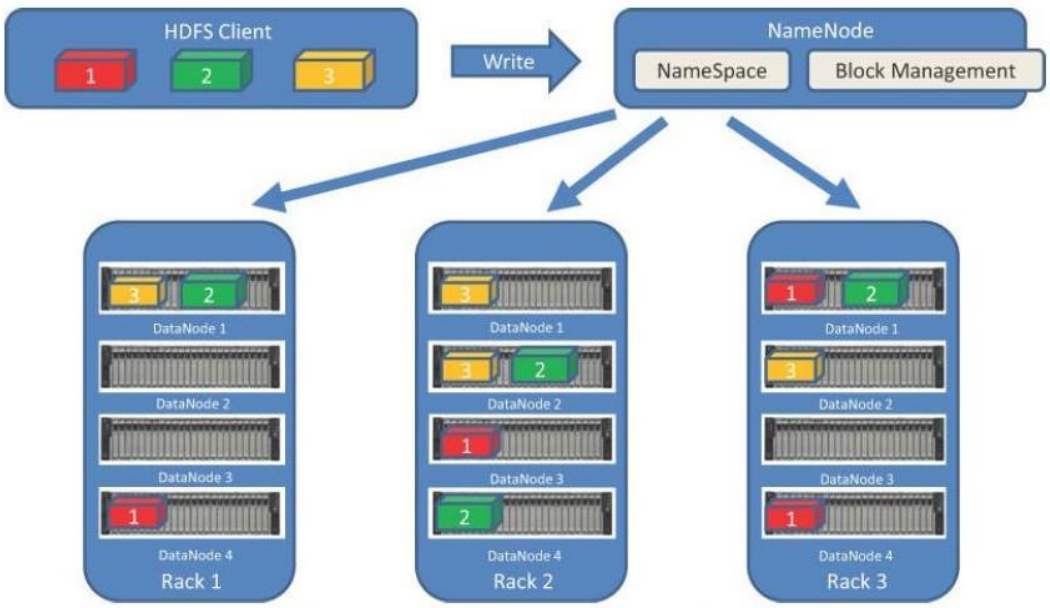


Figure 7-9 *Writing a File to HDFS*

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

YARN

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

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

The Hadoop Ecosystem

As mentioned earlier, Hadoop plays an increasingly big role in the collection, storage, and processing of IoT data due to its highly scalable nature and its ability to work with large volumes of data.

Hadoop now comprises more than 100 software projects under the Hadoop umbrella, capable of nearly every element in the data lifecycle, from collection, to storage, to processing, to analysis and visualization. Each of these individual projects is a unique piece of the overall data management solution. The following sections describe several of these packages and discuss how they are used to collect or process data.

7C EXPLAIN LAMBDA ARCHITECTURE

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

One architecture that is currently being leveraged for this functionality is the Lambda Architecture. Lambda is a data management system that consists of two layers for ingesting data (Batch and Stream) and one layer for providing the combined data (Serving).

These layers allow for the packages discussed previously, like Spark and MapReduce, to operate on the data independently, focusing on the key attributes for which they are designed and optimized.

Data is taken from a message broker, commonly Kafka, and processed by each layer in parallel, and the resulting datais delivered to a data store where additional processing or queries can be run. Figure shows this parallel data flow through the Lambda Architecture.

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

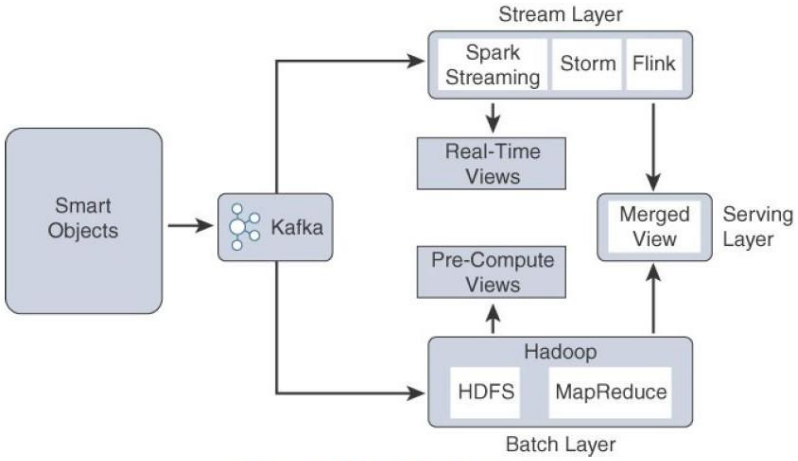


Figure 7-11 *Lambda Architecture*

nicely:

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

8 A EXPLAIN EDGE STREAMING ANALYTICS AND FUNCTIONS OF EDGEANALYTICS PROCESSING UNIT.

One industry where data analytics is used extensively is the world of automobile racing. For

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

Comparing Big Data and Edge Analytics

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

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

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

- **Reducing data at the edge:** The aggregate data generated by IoT devices is generally in proportion to the number of devices. The scale of these devices is likely to be huge, and so is the quantity of data they generate. Passing all this data to the cloud is inefficient and is unnecessarily expensive in terms of bandwidth and network infrastructure.
- **Analysis and response at the edge:** Some data is useful only at the edge (such as a factory control feedback system). In cases such as this, the data is best analyzed and acted upon where it is generated.
- **Time sensitivity:** When timely response to data is required, passing data to the cloud for future processing results in unacceptable latency. Edge analytics allows immediate responses to changing conditions.

Edge Analytics Core Functions

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

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

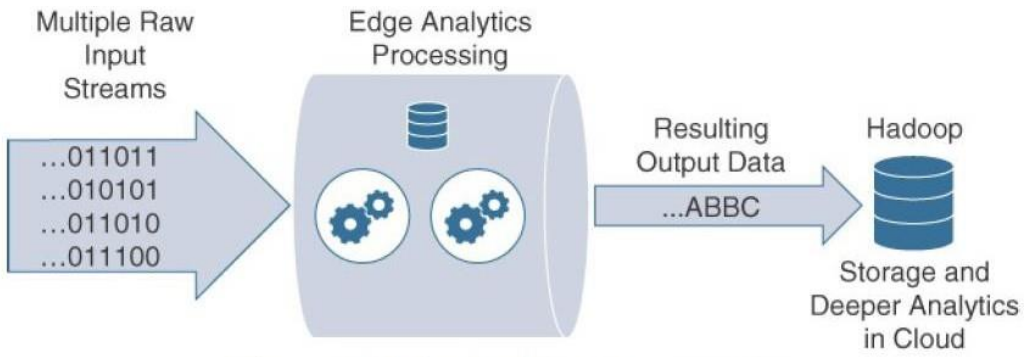


Figure 7-12 *Edge Analytics Processing Unit*

Distributed Analytics Systems

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

back from the edge and look at a wider data set.

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

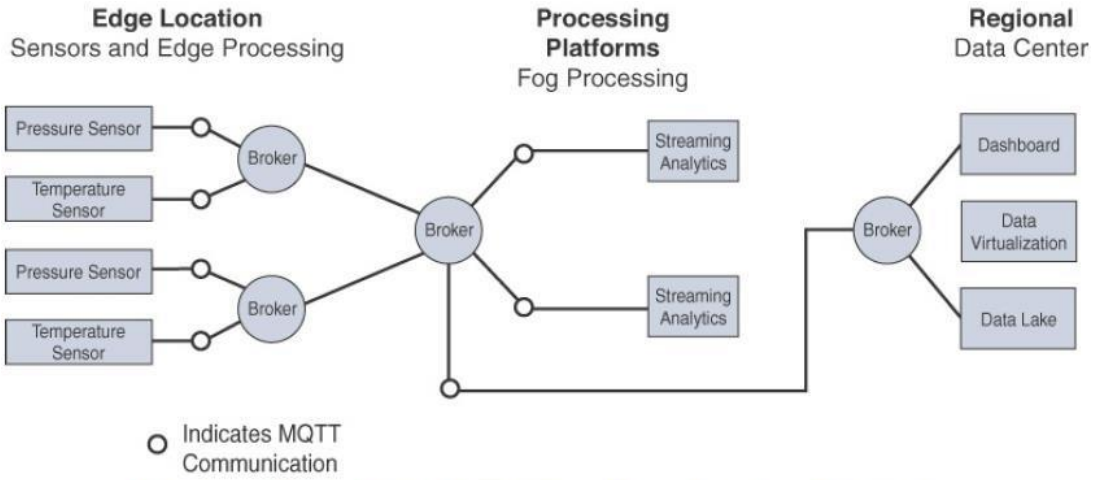


Figure 7-15 *Distributed Analytics Throughout the IoT System*

Network Analytics

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

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

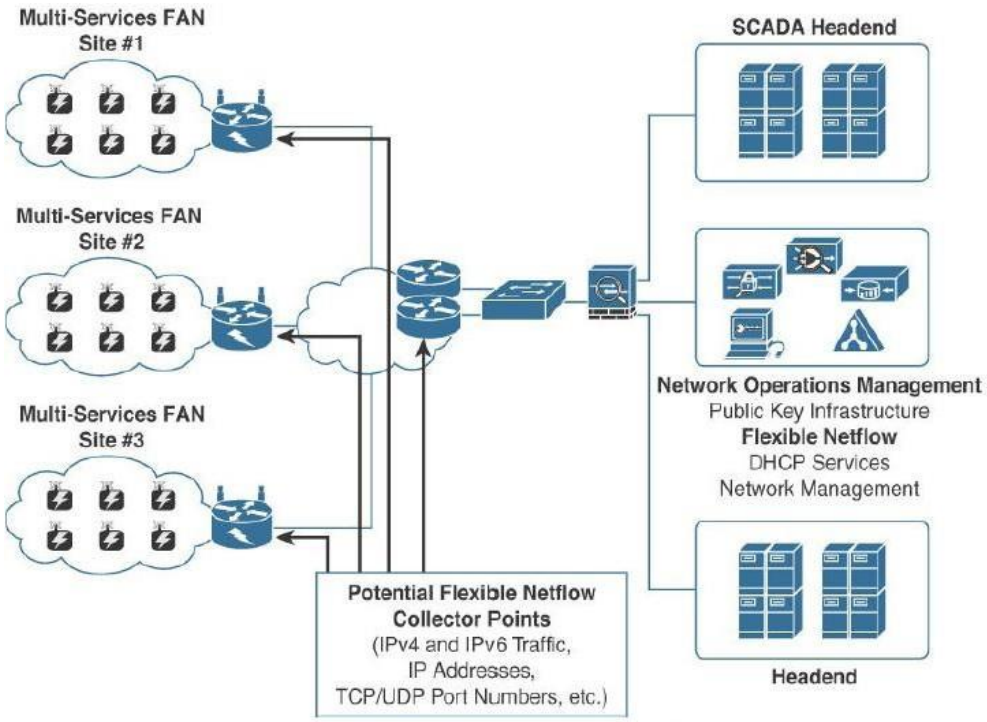


Figure 7-16 *Smart Grid FAN Analytics with NetFlow Example*

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

8B EXPLAIN IN FORMAL RISK ANALYSIS STRUCTURES

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

OCTAVE

OCTAVE (Operationally Critical Threat, Asset and Vulnerability Evaluation) has undergone multiple iterations. The version this section focuses on is OCTAVE Allegro, which is intended to be a lightweight and less burdensome process to implement. Allegro assumes that a robust security team is not on standby or immediately at the ready to initiate a comprehensive security review.

This approach and the assumptions it makes are quite appropriate, given that many operational technology areas are similarly lacking in security-focused human assets. Figure 8-5 illustrates the OCTAVE Allegro steps and phases.

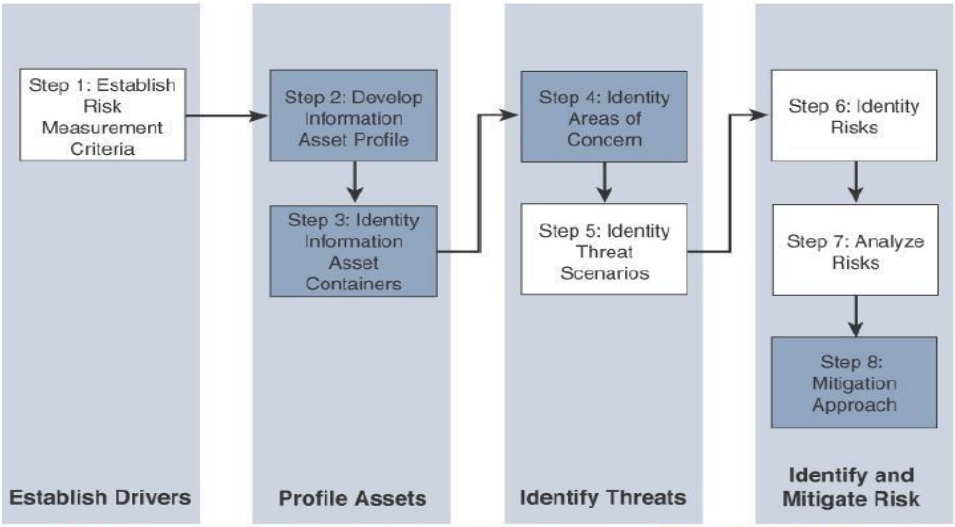


Figure 8-5 *OCTAVE Allegro Steps and Phases (see https://blog.compass-*

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

FAIR

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

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

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

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

9A WRITE NOTES ON: (I) ARDUINO UNO (II)RASPBERRY PI

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

Arduino

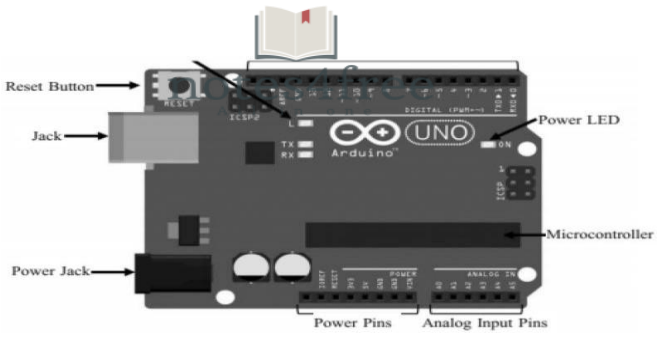
1) Arduino is a open source product , software/hardware which is accessible and flexible to customers.
2) Arduino is flexible because of offering variety of digital and analog pins , SPI (Serial Peripheral Interface) and PWM
3) The **PWM pins** are used for giving the desired analog output .They are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

Arduino

- In the ten years since Arduino was released , hundreds of “Arduino boards” are available in the market serving every kind of purpose.
- We focus on Arduino UNO .
- Some of the Boards from Arduino family are given below.
- Arduino mega is a big sister to the UNO with more memory and pins with a different chip the Atmega2560,useful when your project doesn”t fits in an UNO.
- Arduino Micro is bit smaller with a chip Atmega32u4 that can act like a keyboard or mouse which does its task with native USB.
- Its slim with downward pins which can be plugged into a breadboard.
- The Arduino MKR1000 is a little like an Arduino Micro but has a more powerful 32-bit ATSAM ARM chip and built-in WiFi.
- A great upgrade for when you want to do internet of Things projects.
- Flora is an Arduino compatible from Adafruit which is a round wearable which can be sewed (attach) into clothes.

ARDUINO UNO Learning Board.

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



Arduino Uno Learning Board

Configuring Raspberry Pi

- sudo nano /boot/config.txt
- Add this → “dtoverlay=w1-gpio” at the end of file if doesn’t exist
- sudo reboot

- sudo modprobe w1-gpio
- sudo modprobe w1-therm
- cd /sys/bus/w1/devices
- ls
- cd 28-xxxx
- cat w1_slave
- a3 01 4b 46 7f ff 0e 10 d8 : crc=d8 YES
- a3 01 4b 46 7f ff 0e 10 d8 t=32768

5.10) Pi Via SSH (Secure Shell)

- You can access the command line of a Raspberry Pi remotely from another computer or device
- The Raspberry Pi will act as a remote device: you can connect to it using a client on another machine.

1. Set up your local network and wireless connectivity

- Make sure your Raspberry Pi is properly set up and connected.
- If you are using wireless networking, this can be enabled via the desktop's user interface, or using the command line
- You will need to note down the IP address of your Pi in order to connect to it later.
- Using the ifconfig command will display information about the current network status, including the IP address, or you can use hostname -I to display the IP addresses associated with the device

2. Enable SSH

- Enter sudo raspi-config in a terminal window
- Select Interfacing Options
- Navigate to and select SSH
- Choose Yes
- Select Ok
- Choose Finish

Alternatively, use systemctl to start the service

- sudo systemctl enable ssh
- sudo systemctl start ssh

3. Enable SSH on a headless Raspberry Pi (add file to SD card on another machine)

- For headless setup, SSH can be enabled by placing a file named ssh, without any extension, onto the boot partition of the SD card from another computer.
- When the Pi boots, it looks for the ssh file. If it is found, SSH is enabled and the file is deleted

4. Set up your client

SSH is built into Linux distributions and Mac OS. For Windows and mobile devices,third-party SSH clients are available

9B WITH A NEAT DIAGRAM, EXPLAIN WIRELESS TEMPERATURE MONITORING SYSTEM USING RASPBERRY PI

Getting Started

With your Raspberry Pi turned off, build the circuit as per this diagram.

The DS18B20 is placed into the breadboard so that the **flat side faces you**.

- The black jumper cable goes from GND, which is the third pin down on the right column to the first pin of the DS18B20.
- The yellow jumper cable goes from the fourth pin down on the left column and is connected to the middle pin of the DS18B20.
- The red jumper cable goes from the top left pin of the Raspberry Pi to the far right pin of the DS18B20.

  The Resistor connects the RIGHT pin to the MIDDLE pin. This is called a **pull up resistor** and is used to ensure that the middle pin is always on. In the diagram I had to use a spare red wire to show this connection. But in reality, using the resistor to make the connection, as per this photo is the best way.

Configuring the Raspberry Pi

We now need to take two steps to enable our DS18B20 for use.

Install the Python Library

Firstly we need to install a Python library, pre-written code that enables the Python code that we shall later write to talk to the sensor. The Python library is called **w1thermsensor** and to install

it we need to use the Terminal. You can find the terminal icon in the top left of the screen. It looks like...

When the terminal opens, enter the following to install the library, just press **ENTER** to start.

```
sudo pip3 install w1thermsensor
```

Enable the Interface

The DS18B20 uses a 1 wire serial interface, this is the middle pin of the sensor, that is connected to the Raspberry Pi via the yellow wire in the diagram. We need to tell our Raspberry Pi that we are using this pin and to do that we use the **Raspberry Pi Configuration** tool, found in the **Preferences** menu.

When it opens, click on the **Interfaces** tab and then click on **Enable** for the **1-Wire** interface.

Now click on **Ok** and you will be asked to reboot, so go ahead and do that, and let the Raspberry Pi reboot to the desktop.

Writing the Python Code

Our project will gather the temperature from the DS18B20 sensor every one second and print it to the screen. The code will run forever.

To write the code we shall use the **Python 3 Editor** found in the **Programming** menu.

When the application opens, click on **File >> New** to create a new blank document. In this new window, click on **File >> Save** and call the project **temperature-sensor.py**

Importing the libraries

The first step in any Python project that uses libraries is to **import** the libraries that we wish to use. In this case we import **time** to control how often the sensor data is collected, and we import **w1thermsensor** to enable our project to talk to the sensor.

```
import time

from w1thermsensor import W1ThermSensor
```

Sensor

Our next line is to create an object to store a connection the sensor. So rather than typing `W1ThermSensor()` everytime we want to use the sensor, we store the connection in an object called `sensor`.

```
sensor = W1ThermSensor()
```

Running in a loop

We'd like to get the temperature sensor data every second, and run forever. So let's use a **while True** loop to run the code inside of it forever.

```
while True:
```

Now the next lines of code are **indented**, this is how Python shows that this code belongs inside the loop that we have just created.

The first thing to do in our loop is to get the current temperature from the DS18B20 sensor, and then store it in a variable called **temperature**. Variables are boxes / containers into which we can store any data.

```
    temperature = sensor.get_temperature()
```

Now that we have the data, lets print it to the screen using the `print()` function. But lets use the data in the form of a sentence that will tell the us what the temperature is in celsius. For this we use a little Python trick called **string formatting**, where we would like the temperature data to be printed in the sentence, we use an **%s** which will format the temperature data from an **float** (a number with a decimal place) to a **string** (text, characters that can be printed but not used in any mathematical equations)

```
    print("The temperature is %s celsius" % temperature)
```

Our last line of Python code will tell the Raspberry Pi to wait for 1 second between taking a temperature reading.

```
    time.sleep(1)
```

Complete Code Listing

```
import time

from w1thermsensor import W1ThermSensor
```

```
sensor = W1ThermSensor()



while True:

    temperature = sensor.get_temperature()

    print("The temperature is %s celsius" % temperature)

    time.sleep(1)
```

10A EXPLAIN IOT STRATEGY FOR SMART CITIES

An IoT Strategy for Smarter Cities

- Managing a city bears some resemblance to managing a corporate enterprise.
- As the need for efficiency increases, new tools help increase operational efficiency.
- For cities, just as for businesses, digitization transforms the perspective on operations.
- New ideas emerge, bringing different approaches to solving management issues.

Vertical IoT Needs for Smarter Cities

- There are many differing approaches and solutions for city management.
- Allthese solutions typically start at the street level, with sensors that capture data on everything from parking space availability to water purity.
- Data analytics is also used extensively—for example, to reduce crime or improve traffic flows.
- Citizens can use tools to leverage their smart mobile devices, such as to report problems and make recommendations for improving urban life or locate available parking spaces.
- When enabled through connectivity, these smart solutions can have a transformative impact on quality of life.
- A recent Cisco study, as illustrated in Figure 5.11, expects IoT to have the following economic impact over a 10-year period:

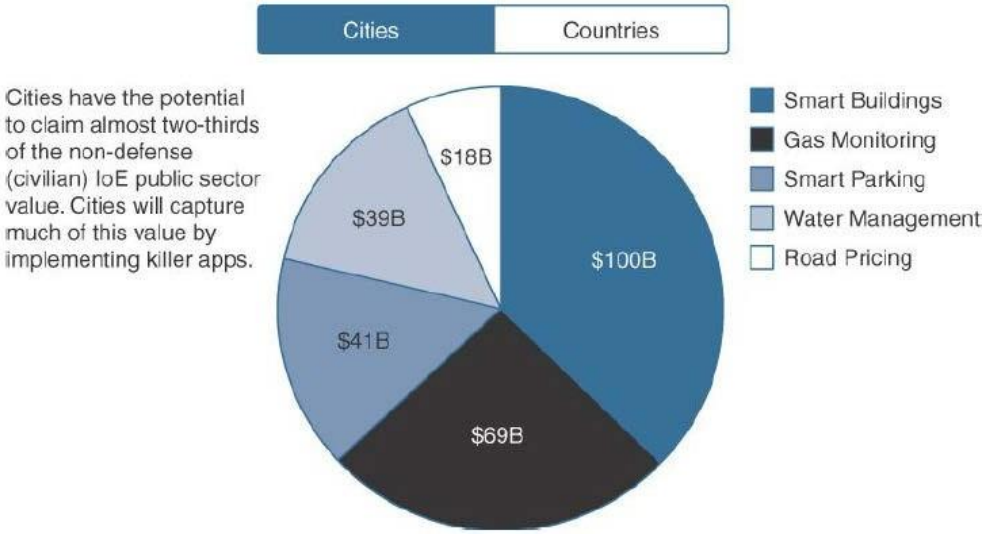


Figure 5.11 Key Use Cases for Smart Cities

1) Smart buildings: Smart buildings have the potential to save $100 billion by lowering operating costs by reducing energy consumption through the efficient integration of heating, ventilation, and air-conditioning (HVAC) and other building infrastructure systems.

- Note that the financial gain applies to city budgets only when a building is city owned.
- However, the reduced emissions benefit the city regardless of who owns the buildings.

2) Gas monitoring: Monitoring gas could save $69 billion by reducing meter-reading costs and increasing the accuracy of readings for citizens and municipal utility agencies.

- The financial benefit is obvious for users and utility companies when the utility is managed by the city.
- There are also very important advantages in terms of safety, regardless of who operates the utility.
- In cases of sudden consumption increase, a timely alert could lead to emergency response teams being dispatched sooner, thus increasing the safety of the urban environment.

3) Smart parking: Smart parking could create $41 billion by providing realtime visibility into parking space availability across a city.

- Residents can identify and reserve the closest available space, traffic wardens can identify noncompliant usage, and municipalities can introduce demand based pricing.

4) Water management: Smart water management could save $39 billion connecting household water meters over an IP network to provide remote usage and status information.

- The benefit is obvious, with features such as real-time consumption visibility and leak detection.
- A gate or a pump can be opened and closed remotely and automatically in real time, based on a variety of flow input and output analytics data.

- Vibrations can be measured to detect and predict potential equipment failures.
- Repair teams can be dispatched proactively before equipment failure occurs.

Road pricing: Cities could create $18 billion in new revenues by implementing automatic payments as vehicles enter busy city zones while improving overall traffic conditions.

- Real-time traffic condition data is very valuable and actionable information that can also be used to proactively reroute public transportation services or private users.

Global vs. Siloed Strategies

- The main obstacle in implementing smart solutions in today's traditional infrastructure is the complexity of how cities are operated, financed, regulated and planned.
- Cities attempting to upgrade their infrastructure to match the growing needs of the citizen population often invest in one problem at a time, and they do it independently.
- Even cities using IoT technology break up city assets and service management into silos that are typically unable to communicate or rely on each other.

The independent investment model results in the following problems:

  - Isolation of infrastructure and IT resources
  - No sharing of intelligence and information, such as video feeds and data from sensors.
  - Waste and duplication in investment and effort
  - Difficulty scaling infrastructure management
- All these requirements pose technological challenges, including the following:
  - How do you collect the data? What are the various sources of data, including hardware endpoints and software?
  - How do you make sure that any data collection devices, such as sensors, can be maintained without high costs?
  - Where do you analyze the data? What data do you carry back to the cloud, and what data do you analyze locally?
  - What kind of network connectivity is best suited for each type of data to collect?
  - What kind of power availability and other infrastructure, such as storage, is required?
  - How do you aggregate data from different sources to create a unified view?
  - How do you publish the data and make it available for applications to consume?
  - How do you make the end analysis available to specialized smart city personnel, such as traffic operators, parking enforcement officers, street lighting operators, and so on at their logical decision points?
  - How do you present the long-term analysis to city planners?

10B WITH NEAT SMART CITIES LAYERÇD ARCHITECTURE DIAGRAM,EXPLAIN SMART CITY IOT ARCHITECTURE.

- A smart city IoT infrastructure is a four-layered architecture, as shown in Figure 5-12.
- Data flows from devices at the street layer to the city network layer and connect to the data center layer, where the data is aggregated, normalized, and virtualized.
- The data center layer provides information to the services layer, which consists of the applications that provide services to the city.

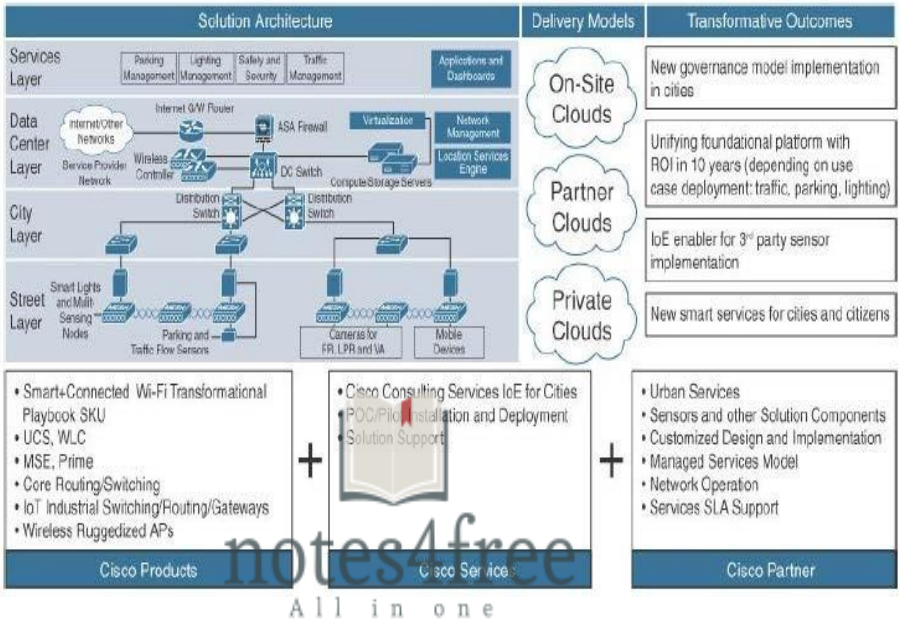


Figure 5.12 Smart Cities Layered Architecture

1) Street Layer

- The street layer is composed of devices and sensors that collect data and take action based on instructions from the overall solution, as well as the networking components needed to aggregate and collect data.
- A sensor is a data source that generates data required to understand the physical world. Sensor devices are able to detect and measure events in the physical world.
- ICT (information and communication technology) connectivity solutions rely on sensors to collect the data from the world around them so that it can be analyzed and used to operationalize use cases for cities.
- A variety of sensors are used at the street layer for a variety of smart city use cases
    - A magnetic sensor can detect a parking event by analyzing changes in the surrounding magnetic field when a heavy metal object, such as a car or a truck, comes close to it (or on top of it).

- A lighting controller can dim and brighten a light based on a combination of time-based and ambient conditions.
- Video cameras combined with video analytics can detect vehicles, faces, and traffic conditions for various traffic and security use cases.
- An air quality sensor can detect and measure gas and particulate matter concentrations to give a hyper-localized perspective on pollution in a given area.
- Device counters give an estimate of the number of devices in the area, which provides a rough idea of the number of vehicles moving or parked in a street or a public parking area, of pedestrians on a sidewalk, or even of birds in public parks or on public monuments—for cities where bird control has become an issue.

2) City Layer

- At the city layer, which is above the street layer, network routers and switches must be deployed to match the size of city data that needs to be transported.
- This layer aggregates all data collected by sensors and the end-node network into a single transport network.
- The city layer may appear to be a simple transport layer between the edge devices and the data center or the Internet.
- However, one key consideration of the city layer is that it needs to transport multiple types of protocols, for multiple types of IoT applications.
- Some applications are delay- and jitter-sensitive, and some other applications require a deterministic approach to frame delivery.
- As a result, the city layer must be built around resiliency, to ensure that a packet coming from a sensor or a gateway will always be forwarded successfully to the headend station. Figure 5.13 shows a common way of achieving this goal.

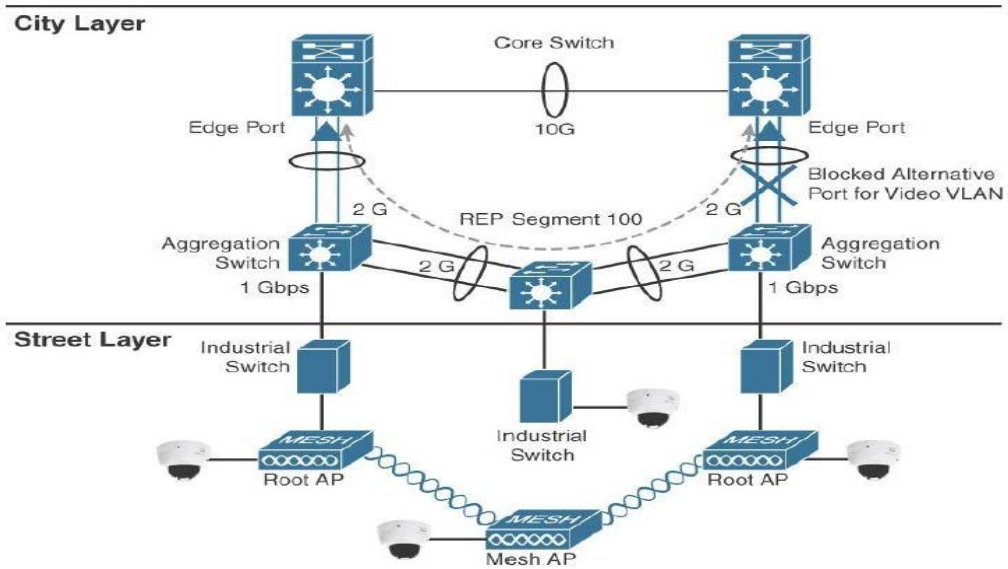


Figure 5.13 Street Layer Resiliency

3) Data Center Layer

- Ultimately, data collected from the sensors is sent to a data center, where it can be processed and correlated.
- Based on this processing of data, meaningful information and trends can be derived, and information can be provided back.
- For example, an application in a data center can provide a global view of the city traffic and help authorities decide on the need for more or less common transport vehicles.
- At the same time, an automated response can be generated.
- For example, the same traffic information can be processed to automatically regulate and coordinate the street light durations at the scale of the entire city to limit traffic congestion.
- The key technology in creating any comprehensive smart solution with services is the cloud.
- With a cloud infrastructure, data is not stored in a data center owned directly or indirectly by city authorities.
- Instead, data is stored in rented logical containers accessed through the Internet.
- Because the containers can be extended or reduced based on needs, the storage size and computing power are flexible and can adapt to changing requirements or budget conditions.
- In addition, multiple contractors can store and process data at the same time, without the complexity of exclusively owned space.
- This proximity and flexibility also facilitate the exchange of information between smart systems and allow for the deployment of new applications that can leverage information from several IoT systems.
- Figure 5.14 shows the vision of utilizing the cloud in smart solutions for cities.
- The cloud provides a scalable, secure, and reliable data processing engine that can handle the immense amount of data passing through it.
- Smart city issues require not just efficient use of infrastructure, which the cloud helps enable, they also require new data processing and management models.
- For example, cloud services allow for Software as a Service (SaaS) models that create cyclical returns on investment.

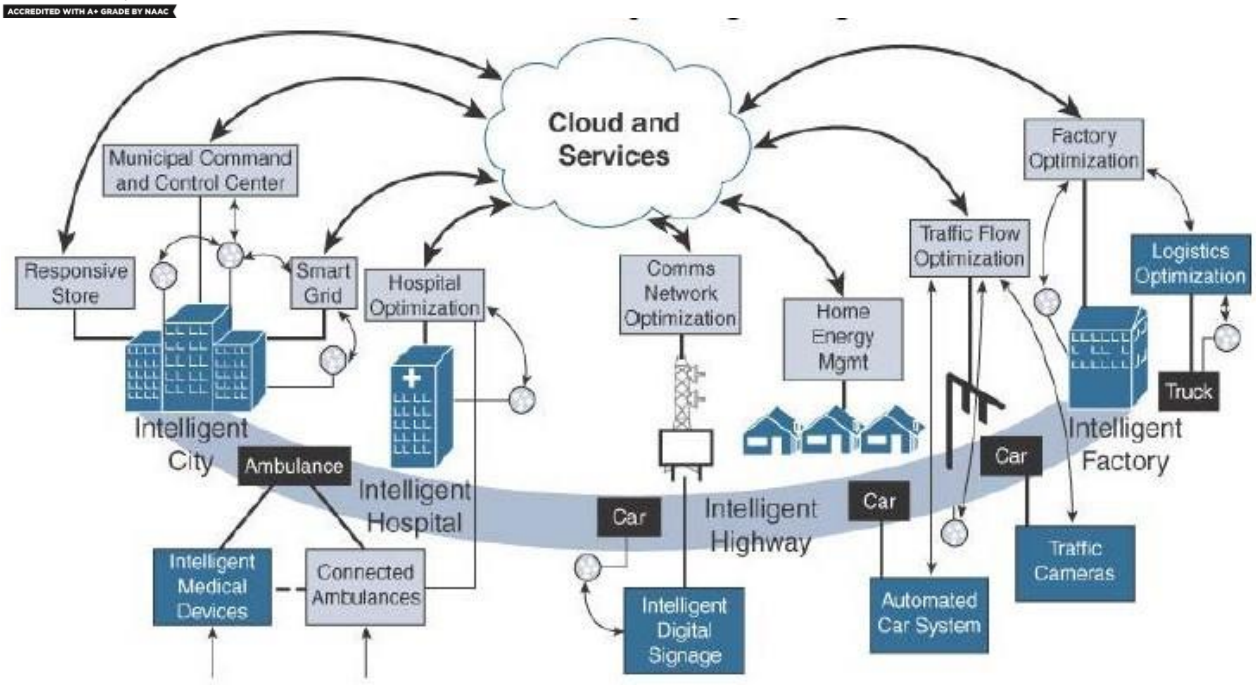


Figure 5.14 The Role of the Cloud for Smart City Applications

- With the cloud approach shown in Figure 5.14 , smart cities can also take advantage of operating expense–based consumption models to overcome any financial hurdles in adopting solutions to their most critical issues.
- Critical data, such as air condition (humidity, temperature, pollution) levels monitoring, can be processed initially.
- Then, as the efficiency of IoT is scaled up, richer data processing can be enabled in the cloud applications.
- For example, the humidity level can be used to regulate the color and luminosity of street lights. In times when city budgets are strained, data processing can be scaled down toessential services.

4) Services Layer

- Ultimately, the true value of ICT connectivity comes from the services that the measureddata can provide to different users operating within a city.
- The collected data should be visualized according to the specific needs of each consumerof that data and the particular user experience requirements and individual use cases.
- For example, parking data indicating which spots are and aren't currently occupied can drive a citizen parking app with a map of available spots, as well as an enforcement

officer's understanding of the state (utilization and payment) of the public parking space, while at the same time helping the city operator's perspective on parking problem areas inthe city at any given time.

- With different levels of granularity and scale, the same data performs three differentfunctions for three different users.
- Along the same lines, traffic information can be used by individual car drivers to findthe least congested route.
- A variation of the same information can be made available to public transportation usersto estimate travel times.
- Public transportation systems, such as buses, can be rerouted around known congestionpoints.
- The number of subway trains can be increased dynamically to respond to an increase in traffic congestion, anticipating the decisions of thousands or even millions of commuters to take public transportation instead of cars on days when roads are very congested.
- Here again, the same type of data is utilized by different types of users in different ways based on their specific use cases.

On-Premises vs. Cloud

- Different cities and regions have different data hosting requirements based on security or legal policies.
- Data can be hosted on-premises or in the cloud. Fog architectures provide an intermediatelayer.
- The data resulting from fog processing can be sent to the cloud or to a data center operated locally (on-premises).
- On-premises encompasses traditional networks, and all their limitations, whereas cloud hosting encompasses a whole host of security risks if the proper measures are not taken tosecure citizen data.
- When data is sent to the cloud, data sovereignty (supremacy) laws may restrict the physical location where this data is actually stored.
- Ideally, a smart city utilizing ICT connectivity would use the cloud in its architecture, but if this is impossible, the city would need to invest far more in the city layer's networking components (for example, switches, routers) and still may not be able to drive the same cross-domain value propositions and scalability in its design.