R Praghatha
T Vinotha
V C Priyadharshini
P Latha
G Subbulakshmi
R Kavitha

Internet of Things

DeepScience

Internet of Things

R Praghatha

Department of Computer Science and Engineering at St. Jospeh's College of Engineering and Technology- Thanjavur

T Vinotha

Department of Computer Science and Engineering at St. Jospeh's College of Engineering and Technology- Thanjavur

V C Priyadharshini

Department of Computer Science and Engineering at St. Jospeh's College of Engineering and Technology- Thanjavur

P Latha

Department of Computer Science and Engineering at St. Jospeh's College of Engineering and Technology- Thanjavur

G Subbulakshmi

Department of Computer Science and Engineering at St. Jospeh's College of Engineering and Technology- Thanjavur

R Kavitha

Department of Computer Science and Engineering at St. Jospeh's College of Engineering and Technology- Thanjavur

DeepScience

Published, marketed, and distributed by:

Deep Science Publishing, 2025
USA | UK | India | Turkey
Reg. No. MH-33-0523625
www.deepscienceresearch.com
editor@deepscienceresearch.com
WhatsApp: +91 7977171947

ISBN: 978-93-7185-983-7

E-ISBN: 978-93-7185-106-0

https://doi.org/10.70593/978-93-7185-106-0

Citation: Praghatha, R., Vinotha, T., Priyadharshini, V. C., Latha, P., Subbulakshmi, G., Kavitha, R. (2025). *Internet of Things*. Deep Science Publishing. https://doi.org/10.70593/978-93-7185-106-0

Preface

First Chapter is about “Basics of IoT And its Architecture” this chapter is contributed by the author Mrs R Praghatha, Assistant Professor in the Department of Computer Science and Engineering at St. Jospeh’s College of Engineering and Technology- Thanjavur.

Second Chapter is about “Design Foundations for Iot-Enabled Devices” this chapter is contributed by the author Mrs T Vinotha, Assistant Professor in the Department of Computer Science and Engineering at St. Jospeh’s College of Engineering and Technology- Thanjavur.

Third Chapter is about “Web Integration and connectivity essentials” this chapter is contributed by the author Mrs V C Priyadharshini, Assistant Professor in the Department of Computer Science and Engineering at St. Jospeh’s College of Engineering and Technology- Thanjavur.

Fourth Chapter is about “Foundations of Internet-Based connectivity systems” this chapter is contributed by the author Mrs P Latha Assistant Professor in the Department of Computer Science and Engineering at St. Jospeh’s College of Engineering and Technology- Thanjavur.

Fifth Chapter is about “Modern Methods For Data Acquisition Analytical processing” this chapter is contributed by the author Mrs G SUBBULAKSHMI Assistant Professor in the Department of Computer Science and Engineering at St. Jospeh’s College of Engineering and Technology- Thanjavur.

Fifth Chapter is about “Smart Data collection and Cloud based computing” this chapter is contributed by the author Mrs R Kavitha, Assistant Professor in the Department of Computer Science and Engineering at St. Jospeh’s College of Engineering and Technology- Thanjavur.

Table of Contents

Deep Science Publishing, 2025
https://doi.org/10.70593/978-93-7185-106-0

Chapter 1: Basics of IoT and Its Architecture

1 Introduction

The Internet of Things (IoT) denotes a paradigm that permits the interaction of interconnected devices and applications, as such that will allow physical objects to communicate with each other through the Internet, which are known as things. The concept first occurred with identity-communication technologies, including Radio Frequency Identification (RFID), in which objects can be tagged with the purpose of future identification and remotely value, observe or manage the item using internet-based mindsets. In the current days, IoT is utilized in a diversity of applications such as tracking via GPS, IoT-monitored gadgets, machine-to-machine (M2M) communication, smart cars, wearable and personal gadget interactions, and other Industry 4.0 projects.

Definition

Internet of Things is a type of network of physical items that are able to transmit, accept, and share information via the Internet or any other type of communication technologies, similarly to the work of computer, tablets and mobile devices. It is a form of connectivity which allows monitoring, coordination and control of processes in different data networks.

IOT vision

The vision of Internet of Things envisages the world where all common objects, wearable gadgets, alarm clocks, home appliances as well as objects around are smarted with integrated sensors, computing and communication technologies. These smart objects communicate with systems, remote ones such as servers, cloud services, applications and services and even people via the Internet, Near-Field Communication (NFC) and other forms of communication.

Example:

Connected to embedded computing, an umbrella can also be changed to a smart device. This is a tiny internal module which provides communication with the owner as well as connection to an online weather service. It takes daily predictions, processes them, and gives alerts towards the commuting time of the owner. Conditions can be indicated in LED displays like the red display of a sunny weather and the yellow display of a rain.

Also possible is the transmission of notifications to the phone of the user through NFC, Bluetooth, and SMS like instructing the user to bring the umbrella on a sunny day or a rainy day. The owner will then have an option of taking the Internet enabled umbrella along.

1.2 IoT Conceptual Framework

The Internet of Things (IoT) has a conceptual framework that is based on a layered architecture that describes the interactions between devices, data and applications. It typically commences physically with data collection in the form of objects, and then there is an edge level processing, network transfer, storage and finally the data analysis to serve the applications and services. In essence, IoT makes any object identifiable, communicative and able to interact via the Internet.

Basic building blocks of IoT conceptual model.

Physical discrete devices and controllers:

The physical things in IoT that do the job of generating data are known as sensors, actuators, and controllers and are referred to as the foundational layer.

Connectivity:

This layer envelops the communication infrastructure that programs the data statistical flow between the devices comprising network interfaces and processing modules.

Edge Computing:

It is also known as data element analysis, in this layer, information goes through initial processing close to the source afterward filtering, transformation, or aggregation of information is done before data is sent to central systems.

Data Accumulation:

This is where raw data is stored by processing and collection which is normally in the cloud-based storage systems.

Data Abstraction:

This layer brings out order, collimates information to provide a common and uniform appearance, which facilitates easier access and utilization by the applications.

Application:

In this case, information is processed and decoded to produce enlightenment. It encompasses the tools and services that would allow users to control, monitor as well as evaluate the performance of systems.

Techniques: The team members have been working collaboratively in group work, which is an important feature.<|human|>Collaboration and Processes:

On the highest level, there is human and organizational work flows. The intelligence offered by the IoT data is incorporated into the decision-making and operational strategies.

Critical aspects of IoT Architectures.

Heterogeneity:

The devices in IoT come in a wide range of format in terms of hardware, software in addition to the networking technologies and therefore the framework should be able to support a flexible and adaptable interaction.

Dynamic Changes:

The framework should support those devices whose states or contexts change with time e.g. powering on/off or moving around and responds to changes in the number of devices in total connection.

Scalability:

A good scheme should be able to accommodate more and more devices, and have the ability to handle the enormous volumes of data that they generate.

Interoperability:

Diversely sourced systems and platforms that belong to different vendors or platforms have to be in a position to communicate and operate cohesively without restriction.

1.3 IoT Architectural View

There are multiple levels (so-called tiers) of an IoT system. A framework is conceptualized using a model, whereas a reference model demonstrates the major

building blocks and their interactions in addition to how they interconnect with each other..

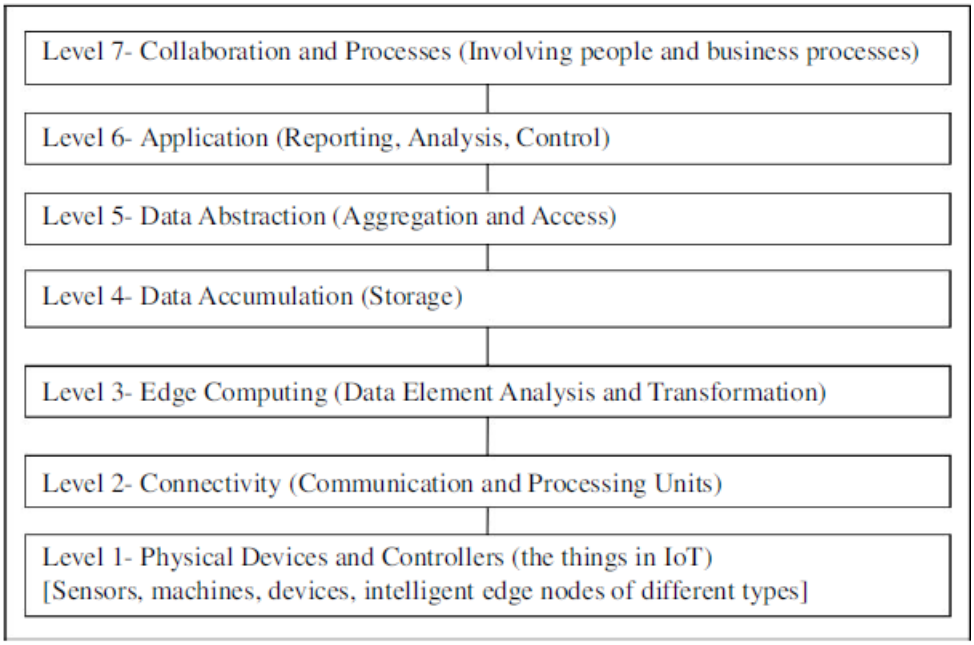


Figure 1: conceptual framework for a general IoT system

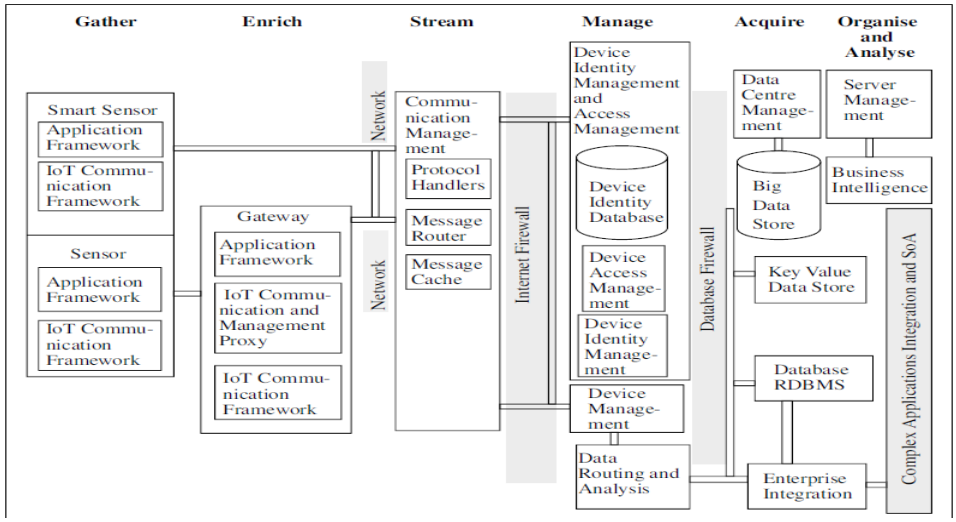


Figure 2: Oracle's IoT architecture

This architecture acts as a model for implementing IoT solutions within service-oriented and business-driven environments.

A group of intelligent sensors collects data, processes and refines it according to the device's application logic, and directly interfaces with a communication manager.

Sensor circuits feed into a gateway equipped with independent functions for data collection, aggregation, computation, and communication; this gateway accepts data in one format and outputs it in a transformed format on the other side.

The communication management layer includes components such as protocol processors, routing modules, and message buffering systems.

It also supports features like maintaining a device identity repository, managing device credentials, and controlling access permissions

The data flows from the gateway, across the Internet and data center infrastructure, to reach the application or enterprise server that processes it.

The organization and analytics layers facilitate service delivery, business process execution, enterprise integration, and advanced processing tasks.

1.4 IoT Technology Fundamentals:

The following components create a rich and varied technological landscape and represent key technologies involved in IoT:

Hardware platforms such as Arduino, Raspberry Pi, Intel Galileo, Intel Edison, ARM mBed, Bosch XDK110, BeagleBone Black, and various wireless SoCs

Integrated Development Environments (IDEs) used to build device-level software, firmware, and application programming interfaces

Communication and application-layer protocols including RPL, CoAP, RESTful HTTP, MQTT, and XMPP (Extensible Messaging and Presence Protocol)

Connectivity technologies like Powerline Ethernet, RFID, NFC, 6LoWPAN, UWB, ZigBee, Bluetooth, Wi-Fi, WiMax, and cellular networks (2G/3G/4G)

Network backbone technologies, for example IPv4, IPv6, UDP, and 6LoWPAN

Software platforms and operating systems such as RIOT OS, Contiki OS, Thingsquare Mist firmware, and Eclipse IoT

Cloud platforms and data-center solutions including Sense, ThingWorx, Nimbits, Xively, openHAB, AWS IoT, IBM Bluemix, Cisco IoT, IOx and Fog, EvryThng, Microsoft Azure, and TCS CUP

Machine learning tools and algorithms, such as GROK by Numenta Inc., which applies machine intelligence to analyze real-time cloud data, detect anomalies, learn continuously, derive actionable insights, and enable advanced automation for streaming-data analysis

Five entities can be considered for the five levels behind an IoT system

Here are **alternate sentence versions** conveying the same meaning more clearly and professionally:

Device platforms include hardware and software built on microcontrollers, system-on-chips, or custom chips, along with the software required for device APIs and web-based applications.

Connectivity and networking components—comprising communication protocols and circuitry—enable devices and physical objects (things) to interconnect and communicate with remote servers over the internet.

Server-side and web development technologies support the creation of web services and web applications that interface with IoT devices.

Cloud platforms provide storage, computing resources, and environments for prototyping as well as developing full IoT products.

Data processing systems, including OLTP, OLAP, data analytics, predictive analytics, and knowledge-discovery tools, support advanced and large-scale IoT applications.

1.4.1Server-Side Technologies

IoT servers encompass application servers, enterprise servers, cloud servers, data centers, and database systems. These servers provide a variety of software capabilities, including:

Online service platforms

- Device identification, identity management, and access control mechanisms
- Functions for collecting, aggregating, integrating, organizing, and analyzing data
- Support for **web applications, web services, and business process execution**

1.4.2 Core Components of an IoT System

Key elements of the IoT devices are:

1 Software embedded into a hardware in the form of a physical object.

2 Hardware Microcontroller, firmware, sensors, control unit, actuators.

3 and communication module.

4 **Communication module**: Software It is comprised of device APIs and device interface. because of communication across the network and communication circuit/port(s), and

5 communication stack creation middleware based on 6LowPAN, CoAP, LWM2M,

6 IPv4, IPv6 and other protocols.

Action programs Software to perform actions, information and commands that the devices.

input and store in the actuators, which allow such things as glowing LEDs,

robotic hand movement etc**.**

Sensors and Control Units

Sensors

Sensors are electronic components designed to detect and measure conditions in the physical environment.

Industrial automation systems or robotic platforms often incorporate multiple intelligent sensors.

Control systems typically rely on sensor–actuator pairs to monitor conditions and trigger appropriate actions.

A smart sensor integrates sensing elements with onboard computing and communication circuitry.

Sensors can be broadly classified into two categories. The first category provides **analog signals** to the control unit. Examples include thermistors, photoconductors, pressure gauges, and Hall-effect sensors.

The second category consists of sensors that deliver **digital signals** to the control unit. Examples include touch sensors, proximity sensors, metal detectors, traffic presence sensors, rotary encoders for angle measurement, and linear encoders for detecting linear displacement.

Control Units

The most widely used control units in IoT devices are Microcontroller Units (MCUs) or specialized custom chips. A microcontroller is essentially an integrated module or core embedded within a VLSI or SoC design. Some common examples include the ATmega328, ATmega32u4, ARM Cortex series and ARM LPC families.

An MCU contains a processor, memory components and multiple hardware peripherals that are interconnected within the chip. It typically includes firmware, timers, interrupt-handling modules and various functional input/output interfaces. Moreover, each MCU family may incorporate circuits tailored for specific applications. For instance, certain versions might feature Analog-to-Digital Converters (ADCs) or Pulse Width Modulation (PWM) units.

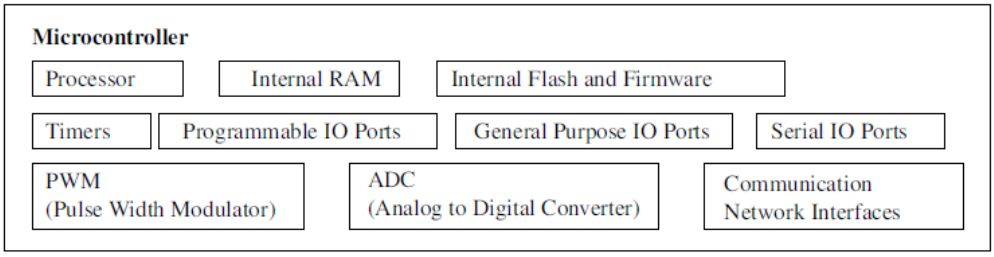


Figure 3 Various functional units in an MCU that are embedded in an IoT device or a physical object

Communication Module

A communication module includes protocol handlers along with a message queue and a message cache. The device's message queue manages messages by adding new ones and removing old ones using a first-in, first-out (FIFO) strategy. The message cache, on the other hand, temporarily stores incoming messages.

The Representational State Transfer (REST) architecture can be applied to enable HTTP-based interactions, using methods such as GET, POST, PUT and DELETE for accessing resources and developing web services.

Software

IoT software is generally divided into two main parts: the software running on the IoT device itself and the software operating on the IoT server.

Middleware

OpenIoT is an open-source middleware platform that supports interaction with sensor clouds and facilitates cloud-based "sensing as a service." Another middleware solution, IoTSyS, provides a communication stack for smart devices, incorporating technologies such as IPv6, oBIX, 6LoWPAN, CoAP and various other standards and protocols. The oBIX framework itself is an XML-based standard and web services protocol known as the Open Building Information Xchange.

Operating Systems (OS)

Some examples of operating systems used in IoT environments include RIOT, Raspbian, AllJoyn, Spark and Contiki.

RIOT is specifically designed for IoT hardware and offers support for a wide range of architectures such as ARM7, Cortex-M0, Cortex-M3, Cortex-M4, standard x86 machines and the TI MSP430 family.

Raspbian, on the other hand, is a widely used OS for Raspberry Pi devices and is built on the Debian Linux distribution.

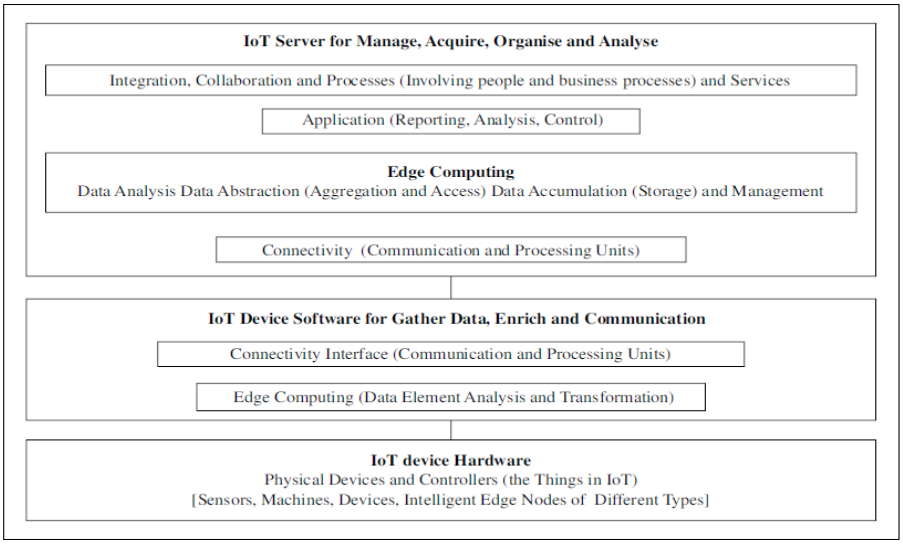


Figure 4IoT software components for device hardware

AllJoyn is an open-source operating system developed by Qualcomm. It is a cross-platform solution that provides APIs for Android, iOS, OS X, Linux and Windows. The platform offers a core framework along with a suite of services, allowing manufacturers to build devices that can easily interoperate.

Spark is a cloud-based, distributed IoT operating system paired with an online integrated development environment. It provides a command-line interface, supports multiple programming languages and includes libraries designed for various IoT hardware platforms.

Contiki OS is an open-source, multitasking operating system tailored for low-power wireless IoT devices. It incorporates essential protocols such as 6LoWPAN, RPL, UDP, DTLS and TCP/IP. It is used in applications like smart city street-lighting systems and operates efficiently with as little as 30 kB of ROM and 10 kB of RAM.

Development Tools and Open-source Frameworks for IoT Implementation

Eclipse IoT (accessible through its official site) offers open-source implementations of several key standards, including MQTT, CoAP, OMA-DM and OMA LWM2M. It also provides tools for working with Lua, along with services and frameworks that support the creation of an Open Internet of Things. Eclipse has also contributed to the development of the Lua programming language. The platform hosts sandbox environments for hands-on experimentation and features live demonstrations. Popular Eclipse IoT projects include Paho, Koneki and Mihini.

Arduino development tools supply a suite of software resources, including an integrated development environment (IDE) and a dedicated programming language aligned with its hardware specifications. These tools enable users to build interactive electronic systems that can sense and manipulate elements of the physical world.

Kinoma offers three major open-source initiatives: Kinoma Create (a prototyping kit), the Kinoma Studio development environment and the Kinoma Platform Runtime. Additionally, Kinoma Connect is a free mobile application available for both iOS and Android, enabling smartphones and tablets to interface with IoT devices.

APIs and Device Interfacing Components

The connectivity interface is made up of communication APIs, hardware device interfaces and processing units.

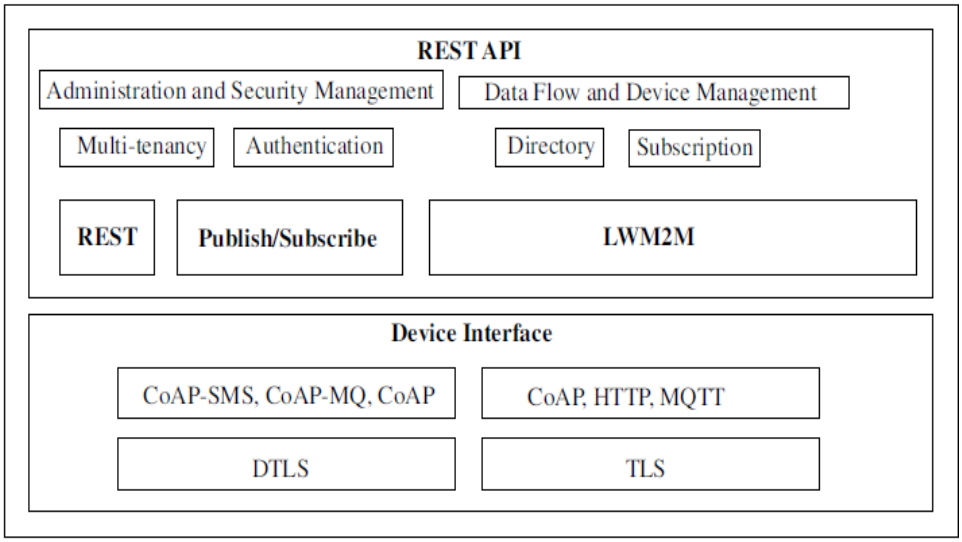


Figure 5mBedTM API and device interfacing components

Platforms and Integration Tools

ThingSpeak is an open data platform that provides a public API for developers. It includes interfaces that support real-time data acquisition, geolocation tracking, data processing and data visualization. The platform allows device status updates and plugin integrations. It can handle HTTP requests as well as store and analyze incoming data. ThingSpeak can work with various hardware and software ecosystems, including Arduino, Raspberry Pi, ioBridge/RealTime.io and Electric Imp. A key advantage of ThingSpeak is its built-in support for MATLAB analytics, along with mobile and web applications and social networking integration.

Nimbits is a cloud-based service compatible with multiple programming languages such as Arduino, JavaScript, HTML and the Nimbits.io Java library. The system can be deployed on Google App Engine, any J2EE server running on Amazon EC2 or on devices like the Raspberry Pi. It is known to handle specific types of data and provides secure data storage. Data that is processed by Nimbits could have timestamps or geolocation marks as well.

The IoT Toolkit offers an API of Smart Object, a semantic layer of HP-to-CoAP and a set of utilities to combine various Ioot sensor networks and communication frameworks.

SiteWhere provides a full system of managing IoT devices. It facilitates the data collection and enables it to integrate with other external platforms. Site Where is compatible with cloud computing at Amazon also available locally. It is also compatible with databases like MongoDB, Apache Hbase and various technologies of big data.

1.5 Sources of IoT

Hardware platforms such as Arduino YUN, Microduino, BeagleBoard and RasWIK are common in creating prototypes of the IoT.. To develop software, firmware and APIs for these hardware prototypes, an integrated development environment (IDE) is typically required.

1.5.1 Popular IoT Development Boards

The Arduino Yún is built around the ATmega32u4 microcontroller, which is compatible with the Arduino ecosystem. It comes with built-in Wi-Fi, Ethernet, a USB port, a micro-SD slot and three reset buttons. The board also integrates an Atheros AR9331 processor capable of running Linux.

Microduino

Microduino is a compact, Arduino-compatible board designed to be stacked with other modular boards. All of its hardware schematics and designs are available as open source.

Intel Galileo

Intel Galileo represents a series of Arduino-certified development boards built on Intel's x86 architecture. As open-source hardware, it features the Intel Quark SoC X1000. Galileo maintains pin compatibility with Arduino boards and offers 20 digital I/O pins (12 native GPIOs), 12-bit PWM for finer control, six analog inputs and support for power over Ethernet (PoE).

Intel Edison

Intel Edison is a compact compute module designed for rapid prototyping and fast development of IoT and wearable applications. It supports smooth device-

to-device communication as well as cloud connectivity. Edison comes with a set of foundational tools for collecting, storing and processing cloud data, applying rule-based logic and generating alerts or triggers using advanced analytics.

Beagle Board

Beagle Board's Beagle Bone series features boards with low power consumption and offers a credit-card-sized computer capable of running Android or Linux. Both its hardware and software are released as open source, making it ideal for IoT development.

Raspberry Pi Wireless Inventors Kit (RasWIK)

RasWIK equips a Raspberry Pi with Wi-Fi connectivity for building wireless IoT projects. It includes documentation for 29 ready-to-build projects, although users may also design their own. While the hardware requires purchase, all provided code is open source and may be used for commercial applications

1.5.2 Role of RFID and IoT Applications

Earlier implementations of IoT often relied on RFID systems connected to the internet. RFID technology supports functions such as tracking, inventory management, supply chain identification, building access control, toll collection and secure entry systems. It is also used in devices like RFID-enabled temperature sensors. Modern RFID networks now play a role in advanced applications including factory automation, third-party logistics (3PL) management, brand authentication and anti-counterfeiting, as well as emerging business processes involving payments, leasing, insurance and quality assurance.

1.5.3 Wireless Sensor Networks (WSNs)

Sensors can be interconnected through wireless technologies, allowing them to work together to observe and measure physical or environmental parameters. These would be sensors that can feed information on areas that might be remote or inaccessible. There is a radio-frequency transceiver on each wireless sensor which allows it to communicate. A sensor node may either have a digital sensor or an analog sensor with a signal-conditioning circuit. These kinds of network are able to keep track of factors such as temperature, lights, darkness, metal distance, movement of traffic and other physical, chemical and biological cues.

WSN Definition

A Wireless Sensor Network (WSN) is a network whereby individual sensor devices are connected and can compute, with options of compressing, aggregating and analyzing data, communicating and transmitting data. The WSN nodes are autonomous i.e. each has its own processing power as well as can make its request and get its replies and perform data forwarding and route finding operations.

A single internet definition covers a WSN as being a wireless network consisting of spatially distributed autonomous devices, which rely on sensors to measure physical or environmental conditions, i.e. temperature, sound, vibration, pressure, motion or pollutants, in multiple locations..”

WSN Node

A sensor node in a WSN typically has limited processing resources. The network topology can change frequently, causing the system to operate similarly to an ad-hoc network. In such dynamic environments, a WSN is usually capable of self-configuration, self-organization, self-healing and self-discovery.

1.6 M2M Communication

Machine-to-machine (M2M) communication refers to the exchange of information between physical devices or machines of similar types, primarily for monitoring but also for control functions. Each machine in an M2M system contains an embedded smart device that senses machine data or status and performs necessary processing and communication tasks. These devices can communicate through wired or wireless channels using protocols such as 6LoWPAN, LWM2M, MQTT and XMPP. Every communication device is typically assigned a 48-bit IPv6 address.

1.6.1 M2M to IoT

In industrial environments, IoT technology brings together advanced physical machinery, M2M communication and networks of sensors, enhanced by analytics, machine learning and knowledge discovery tools.

M2M technology becomes closely aligned with IoT when smart machines gather data and transmit it over the Internet to other devices or remotely located systems.

The key distinction between M2M and IoT is that M2M systems focus on direct device-to-device communication—handling monitoring, coordination and control without requiring the Internet—while IoT systems rely on the Internet, cloud servers, standard Internet protocols and cloud-based applications or services.

M2M applications span various domains including industrial automation, logistics, smart grids, smart cities, healthcare and defense. Originally, M2M was mainly used for automation and instrumentation, but its applications have expanded to include telemetry and the broader Industrial Internet of Things (IIoT).

1.6.2 M2M Architecture

The architecture of an M2M system is divided into three main domains

M2M device domain

M2M network domain

M2M application domain

The M2M device communication domain includes three key elements: the physical devices themselves, the communication interface and the gateway. The communication interface serves as a port or subsystem that receives input from one side and forwards the data to the other.

The M2M network domain incorporates components such as the M2M server, device identity management, data analytics and device/data management—mirroring functions found in IoT architectures, which follow the connect → collect → assemble → analyse flow.

The M2M application domain hosts applications responsible for providing services, monitoring systems, performing analysis and managing or controlling device networks.

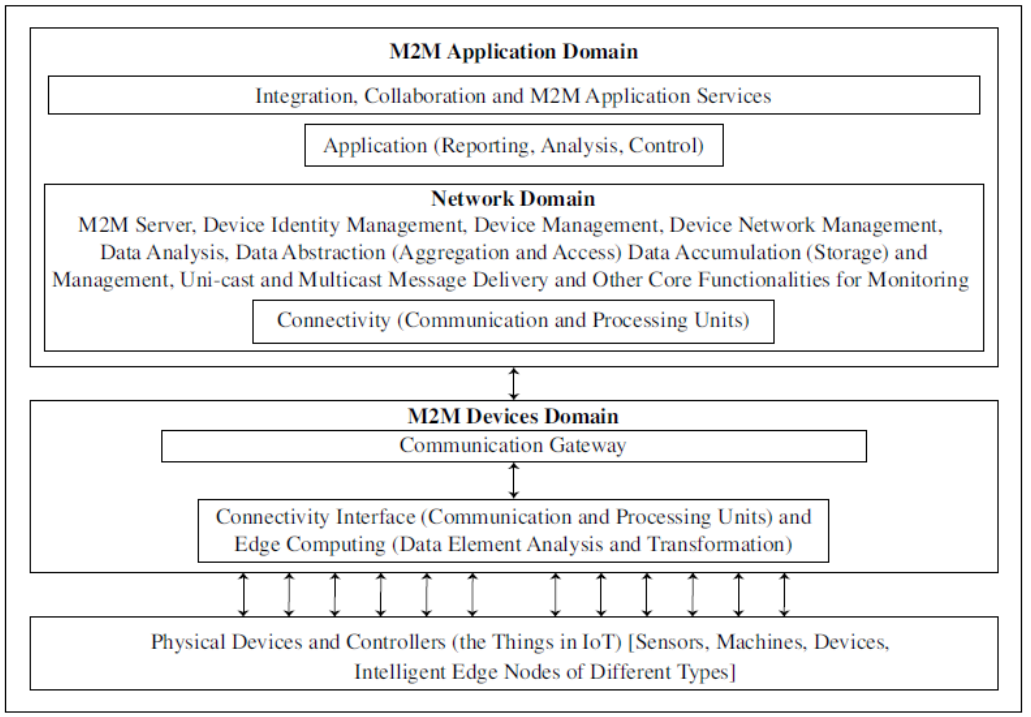


Figure 6 Three domains of M2M architecture

The **M2M device communication domain** includes three main components: the physical devices, the communication interface and the gateway. The communication interface acts as a port or subsystem that accepts input from one side and forwards the data to the other.

The **M2M network domain** comprises the M2M server, device identity management, data analytics and both data and device management. Its structure is comparable to the IoT architecture, following the stages of connecting, collecting, assembling and analyzing data.

The **M2M application domain** contains the applications responsible for delivering services, monitoring systems, performing analysis and managing or controlling device networks.

1.7 Summary

The chapter gives a specific background to the concept of IoT which includes architecture, components of devices, communication technologies, development platform, middleware, cloud tools and application areas. It emphasizes how RFID and M2M systems have evolved to the current IoT solutions which incorporate machine learning, cloud computing and big sensor networks

Deep Science Publishing, 2025
https://doi.org/10.70593/978-93-7185-106-0

Chapter 2: Design Foundations for IoT-Enabled Devices

2.1 Introduction

In IoT/M2M devices, data can be described as the information passed to the applications, services, or business processes and the information passed out to the device to be observed or triggered by an actuator.

The data stack indicates the data that has been accrued at the end of the processing in the levels of intermediary or domains. These layers occur in the OSI model application, presentation, session, transport, network, data link and physical.

KEY TERMS

- A layer is an event of a sequence of operations, each of which has a given protocol or rule to be followed, and the results of the process are passed to the next layer till the whole process is finished.
- One layer is a functional step in organized series of actions whereby the processing of information takes place on a defined approach and then the information flows up or down.Layers may include several sublayers that handle more specialized tasks within the broader operation.
- In a layered model, each layer performs distinct activities and passes results to adjacent layers for further processing.

2. Physical Layer

- The physical layer occurs to process physiological data bits at the transmitting end or the receiving end.
- It employs hard-based schemes such as wired or wireless to transfer bits over the communication medium.
- It is the lowest in the communication models and hence touches electrical, optical or radio signals

- This layer dictates the physical transmission of bits and in turn, the form of cables, frequencies, connectors and signalling.

3. Application Layer

- Application layer will handle the preparation of application generated data to be sent and delivery of incoming data back to applications.
- The information produced by the applications at this layer is passed downwards through a number of other layers to the physical layer where they are transmitted.
- At the receiver end, the physical layer data flows vertically through the various layers till the data reaches the application layer.
- This layer offers interfaces and protocols that the software applications through which the software applications communicate with each other over the network.

4. Level

- A level refers to a level within hierarchy with the simplest functions at the lowest end and the most complicated functions at the top side.
- Levels can be lower e.g. raw data gathering or device behaviors, and high levels are on e.g. analytics or business processes.
- Levels will be used to give simple device interactions up to the high level enterprise interactions.
- They are a hierarchical development between job roles of base activities to decision-making roles that are strategic.

5. Domain

- A domain refers to a collection of components, layers or levels of software having particular functions and capabilities..
- For instance, access networks, CoRE networks, application services, and service platforms together form a network domain.
- A domain typically performs focused tasks and interacts with other domains only to a limited extent.
- Domains act as logical boundaries where related operations are grouped for easier management and organization.

6. Gateway

A gateway is software that links the application layer of the sender with the application layer of the receiver, functioning as an Application Layer Gateway (ALG).

IP

IP refers to Internet Protocol, available in two main versions: IPv6 (version 6) and IPv4 (version 4), both operating at the network layer.

Header

- A header is a collection of octets that stores metadata about the data being transmitted.
- Headers encapsulate the information of a layer before passing it to the next layer during communication between two end systems.
- A header structure and size is determined by the protocol that is being used to construct the data stack in that layer.
- The control information, which includes addresses, sequence numbers and flags, is placed at the beginning and before the actual data is sent over.
- Packet
- The packet is the formatted stack of data which is transmitted by the network in the communication.
- It is a data package and related management data ready to be transmitted by networks as required by the protocols.
- Packet size is derived by application of protocol, it should comply with network-layer principles.
- The encapsulatory data are transferred in a packet along the routing path, and they comply with the size restrictions of the communication protocol used.
- Protocol Data Unit
- Protocol Data Unit (PDU) is the form of data defined within a protocol of a certain layer and transmitted between the neighboring layers.
- Maximal Transmission Unit (MTU)
- MTU- This is the largest size of frame packet or segment that can be sent within networks such as the internet expressed in octets per packet (1 byte = 8 bits) one octet = 1 amino acid.
- The MTU is an acronym used to describe the maximum characterized unit of data which can be transferred in a single frame or packet via a particular network.
- MTU is the peak size of data which can be encapsulated and forwarded without being broken up in network transmitted as packets or in frame networks.

Star Network

- A star network is used to refer to a system in which several nodes interact with a coordinator or master node..
- In a star topology, all devices connect to a single central point for data exchange.
- Star networks involve endpoints linking directly to a hub-like coordinator.

Mesh Network

- A mesh network is made up of nodes which can have interconnections among them creating several routes of communication channels.
- Mesh topology The nodes in mesh topology can be in either one-to-one or one-to-many communication.
- A mesh network enables devices to connect with the various neighbouring nodes to enable them to be routed flexibly..

End-point Device / Node

- An end-point device is a node that connects to a coordinator or router for communication.
- End devices form the leaf nodes of the network, linking upward to coordinators or routers.
- These nodes act as the final points for data generation or reception in the topology.

Coordinator — Related Sentences

- A coordinator is responsible for connecting multiple end-points and routers in a star topology and forwarding data between them.
- The coordinator manages communication among attached nodes and routes data stacks to the correct destination.
- It serves as the central node that oversees network operation and data transfer among devices.

Master — Related Sentences

- A master is the master device, which triggers the connections with other devices in a star topology network.
- The node that initiates the connection process is the master node and forms network control.
- It follows suit in co-ordination of communication set up with other nodes.

Slave -- Related Sentences

- A slave device connects itself to a master, synchronizes itself with the clock of the master and adds the address given by the master.
- Slave nodes rely on the master when operating in the networks in terms of timing and addressing.

- These appliances react to master directives and execute according to rules which are specified by the master.

Router -- Related Sentences

- A router is a gadget that holds information on the routing process and reroutes the information based on the best available route.
- It has logical connections with the destinations and decides the most suitable paths to be taken when relocating data stacks.
- Part of the routing work is performed by routers which make decisions using the stored network paths.

ISM Band -- Related Sentences

- ISM bands are radio bands used in case of Industrial, Scientific, and Medical purposes.
- Frequencies in use as common ISM bands are 2.4 GE all over the world, 915 MHz in North America, 868 MHz in Europe, and 433 MHz in most parts of Asia.
- These are frequency ranges that are extensively utilised in the unlicensed wireless communication technologies.

Application Related Sentences.

- An application is software that is developed to execute a particular task like streetlights monitoring or controlling.
- The applications present task-oriented services and operations based on the requirements of either users or the system.
- It can be defined as software that can run specified operations in an IoT environmen.

Service

- A service represents application software that fulfills a particular role, e.g. generating some reports or displaying charts.
- Services are software modules that provide defined capabilities like data reporting or graphical presentation.
- A service is an application component that delivers functionality—for example, producing charts or creating analytical reports.
- Service software carries out specialized tasks that support user or system needs, such as visualization or report creation.

Process

- A process is a unit of software, which accepts input, carries out functions, which can be a data acquisition or data analysis and then returns output.
- A process is an executed program that uses data to produce some result according to the stipulated logic.
- Each process together with its memory, execution condition, and other resources are handled by the operating system.
- Processes perform certain category of computational operations and the OS manages the scheduling of the activities, memory requirements and control qualities..

2.2 Design Standards and Layered Architecture for IoT/M2M Systems

IoT Design Standardisation Organisations International.

A number of international organisations have played a role in the standardisation of the IoT design and architecture. These consist of the following:

There are several international organizations that are actively engaged in the process of establishing IoT systems, layers, communication, and services standards.

1. Internet Engineering Task Force (IETF) — Related Sentences

- The IETF, a global standards organisation, has initiated efforts to define engineering guidelines and recommendations for Internet of Things systems.
- IETF works on specifications related to IoT layers and engineering requirements for communication, networking, and application functionalities.
- As an international engineering body, the IETF provides standards for protocols, layer structures, and operational aspects of IoT communications.

2. International Telecommunication Union – Telecommunication Standardization Sector (ITU-T) — Related Sentences

- ITU-T has proposed a reference architecture outlining IoT domains, network capabilities, and transport mechanisms to support IoT services and applications.
- ITU-T offers a framework for IoT operations at the application and application-support layers.
- The organisation defines guidelines for IoT transport, network features, and domain-level interactions.

3. European Telecommunication Standards Institute (ETSI) — Related Sentences

- ETSI has worked on establishing standards for communication between machines (M2M), covering devices, gateways, and network domains.
- It developed a high-level architecture for service capabilities and applications in M2M environments.
- ETSI's initiatives focus on creating unified standards to support device-to-device and device-to-gateway communication.

4. Open Geospatial Consortium (OGC) — Related Sentences

- The OGC, an international industry consortium, proposes open standards for sensor technologies, including discovery, capability descriptions, and quality parameters.
- OGC also supports standards for integrating sensor information with geospatial web services.
- The consortium promotes interoperable specifications for sensor data within geographic information systems (GIS).

Adapted OSI Architecture for IoT and M2M Systems

- OSI protocols refer to a group of information-exchange standards jointly developed by ISO and ITU-T. The seven-layer OSI model serves as a universal reference model for designing communication networks.
- The OSI model provides a standardized, seven-layer framework that outlines how communication should occur in a network, and its protocols are created through collaboration between ISO and ITU-T.
- Many data-exchange architectures use the OSI model as a foundation and simplify or adapt its layers based on system requirements.
- Various types of communication models utilize the OSI layer structure except they make changes to comply with design requirements or to simplify the models.
- In the examples of IoT and M2M, the IETF offers modified instances of the OSI model which are more suited to the needs of the lightweight, constrained as well as scalable communication..
- The OSI family of protocols provides a fundamental structure for networking, and various organizations—including IETF—customize this structure for IoT/M2M communication scenarios.

2.2.1 Extended OSI Layer Structure for IoT and M2M Systems

- Data flows from the device side to the application side, and each layer processes the incoming information to form a new data stack before passing it to the next layer.
- As data moves upward, every intermediate layer performs its specific processing tasks and forwards the modified data stack to the subsequent layer.
- The transformation of data occurs across all layers between the lowest functional layer and the highest application layer.
- Similarly, data sent from an application or service travels downward through the intermediate layers before reaching the device end.
- At each stage, the layer adds or interprets information, ensuring that the data progresses correctly from devices to applications and back.
- Device nodes also receive data originating from applications or services, after that data has been processed by the layers in between.

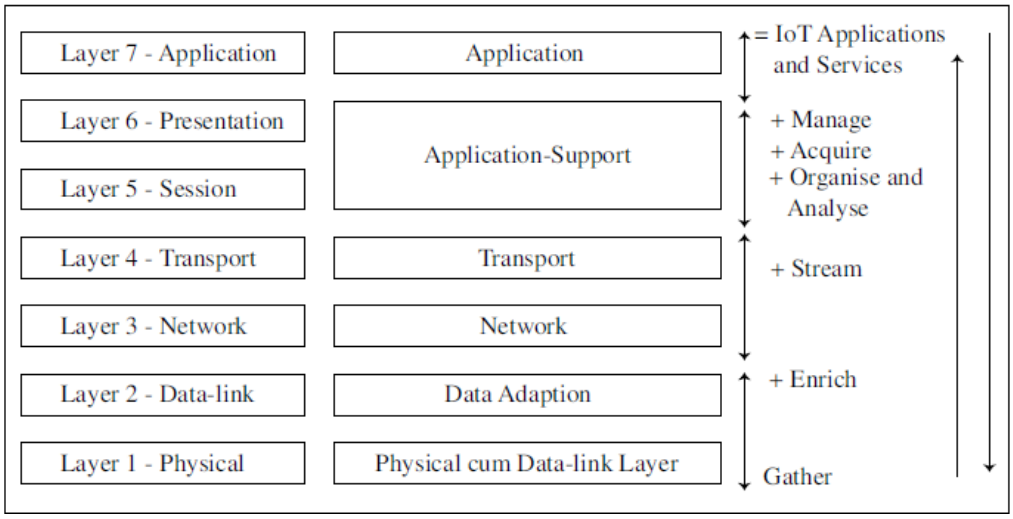


Figure 7Seven-layer generalised OSI model

2.2.2 ITU-T Standard Framework for IoT Operations

- The model also illustrates how its layers correspond to the six-layer modified OSI structure.
- In this model, Layer 1 (L1) represents the device layer, which includes both device and gateway functions.
- Layer 2 (L2) provides network and transport capabilities.
- Layer 3 (L3) functions as the service and application-support layer, offering both generic and specific support capabilities.

- Layer 4 (L4) is the highest layer and contains the applications and services.
- ITU-T defines a four-layer reference model, each layer having distinct roles and capabilities.
- The ITU-T RM1 can be mapped to the modified six-layer OSI model as follows:
- The RM1 device layer corresponds to the OSI data-adaptation layer and the combined physical/data-link layers.
- The RM1 network layer aligns with the OSI network and transport layers.
- The RM1 upper layers match the top two layers of the six-layer OSI model.
- A comparison of ITU-T RM1 with the CISCO IoT Reference Model (RM2) yields:
- RM1's L4 functions align with RM2's collaboration, process, and application levels.
- RM1's L3 support capabilities correspond to RM2's mid-level functions such as data abstraction, accumulation, analysis, and transformation.
- RM1's L2 layer matches RM2's connectivity-level functionalities.
- RM1's L1 device layer aligns with RM2's physical device-level operations.

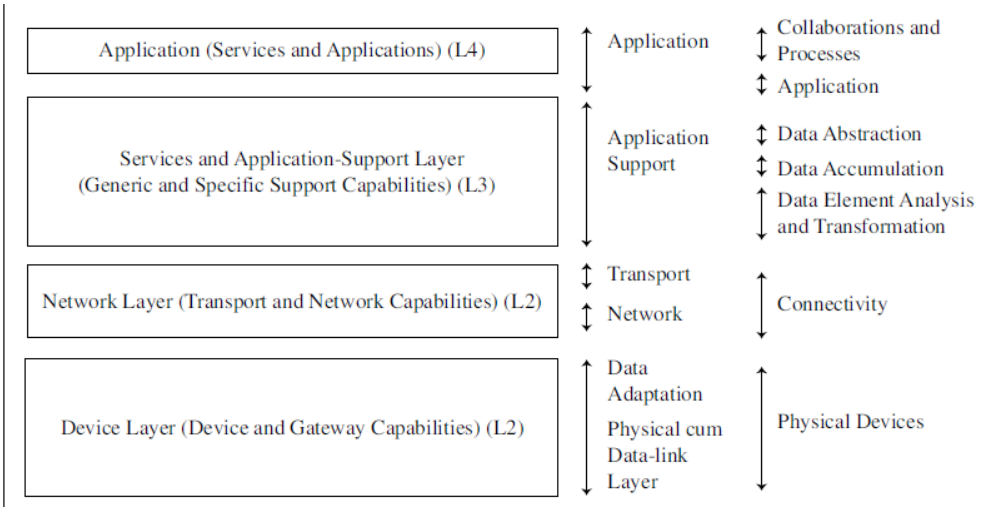


Figure 8ITU-T reference model

2.2.3 ETSI M2M Domains and High-level Capabilities

ETSI Network Domain — Related / Rephrased Sentences

The ETSI network domain includes six major capabilities:

1 M2M applications, which run on top of the communication framework.
2 M2M service capabilities, providing shared functions needed by applications.

3 M2M management functions, responsible for monitoring and controlling devices and services.
4 Network management functions, which handle the operation and maintenance of the underlying communication network.
5 Core network elements, such as 3G, IP networks, control functions, and inter-network connections.
6 Access networks, including LPWAN, Wi-Fi (WLAN), and WiMax technologies.
    - ETSI defines these capabilities to support end-to-end machine communication across diverse networks.

ETSI Device and Gateway Domain

The ETSI device/gateway domain consists of several functional blocks:

- A gateway that connects the M2M area network to the Core and access networks, and also provides M2M service capabilities and application support.
- An M2M area network, which may use technologies such as Bluetooth, ZigBee, NFC, PAN, or LAN.
- M2M devices, which operate within the area network and communicate through the gateway.
- This domain integrates local device communication with broader network connectivity through gateways.

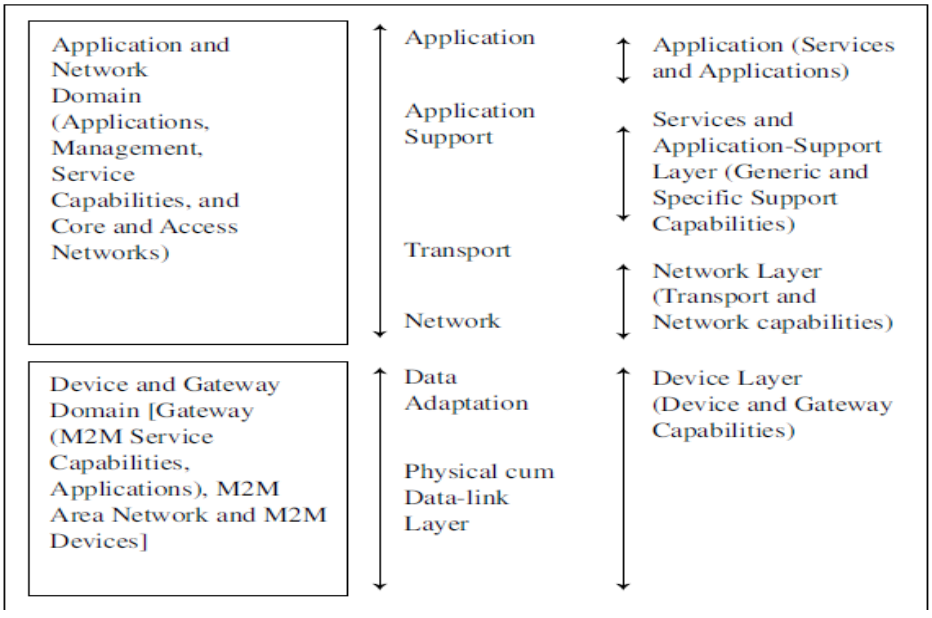


Figure 9ETSI M2M domain architecture

2.3 Communication Technologies and Protocols

The physical and data-link layers comprise a local area or personal area network within the model. In IoT or M2M systems, the local network at these layers can use either wired or wireless communication technologies.

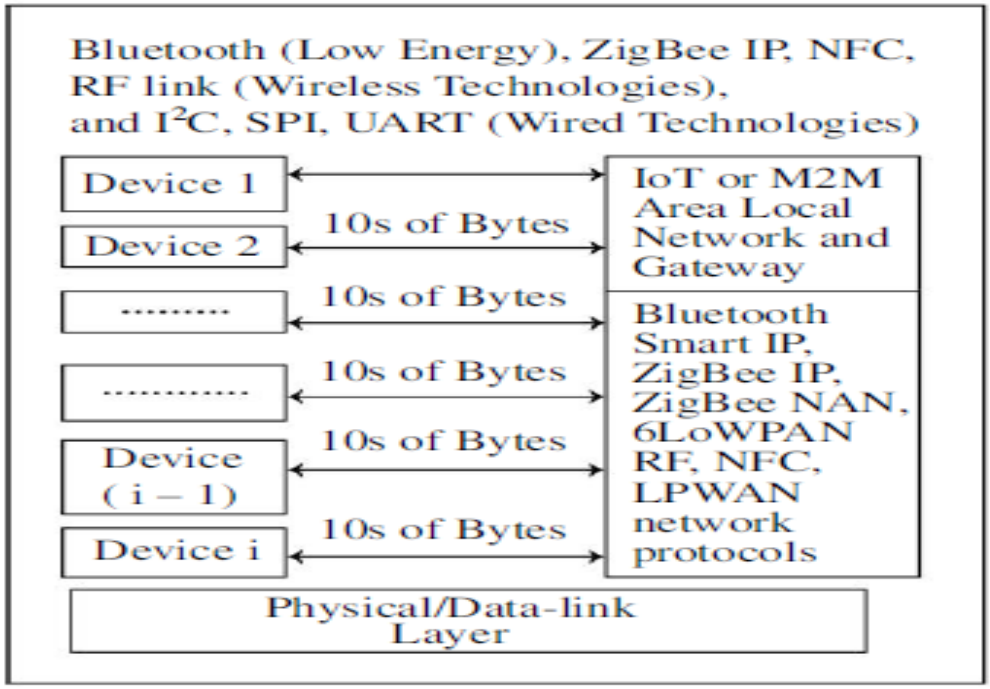


Figure 10LPWAN network protocols

This network connectivity between the devices (left-hand side) has been achieved through either RF, Bluetooth Smart Energy, ZigBee IP, ZigBee NAN (neighbourhood area network), NFC or 6LoWPAN or mobile.

The instance of communications between the device and local devices network is tens of bytes.

Overview of Wireless Communication Technologies

- The physical and data-link layers may rely on either wired or wireless technologies to enable device communication.
- Several short-range wireless technologies—such as NFC, RFID, ZigBee, Bluetooth, and RF transceivers—are commonly used at this level.
- Examples of wireless standards at the physical/data-link layers include NFC, RFID, ZigBee, Bluetooth, and various RF-based modules.

Near-Field Communication in IoT Systems

- Near-Field Communication (NFC) is built upon the ISO/IEC 214443 standard used for contactless proximity cards. NFC functions as a short-range wireless communication method, typically within 20 cm, allowing data transfer between nearby cards and devices.
- It supports applications such as contactless card readers, RFID-based systems, IoT and M2M devices, mobile wallets, smart access keys for vehicles and buildings, and biometric passport verification systems.
- NFC-enabled devices are capable of simultaneously sending and receiving information, with a very fast setup time of around 0.1 seconds. An NFC reader or device can create an RF field to power nearby passive elements like passive RFID tags.
- The NFC modules are able to detect the available RF fields and detect any collision in the signal being relayed to facilitate good communication.
- NFC devices are generally used over a distance of 10-20 cm. NFC enabled devices can connect to Bluetooth or Wi-Fi modules meaning that communication can be increased to almost 30 cm or longer. NFC interface can store the information and consequently transfer same via Bluetooth, LAN, or Wi-Fi networks through conventional handover processes.
- The supported data transfer rates are 106 kbps, 212 kbps, 424 kbps and 848 kbps with kbps representing kilobits/sec.

Communication Modes

1. Point-to-Point (P2P) Mode

- Both the devices in this mode will be active units and send out RF fields to each other alternately and subsequently exchange data.
- P2P enables two devices equipped with an NFC to exchange information such as contacts, files or credentials.

2. Card-Emulation Mode

- The NFC device is similar to an uncontacted smart-card, which allows it to conduct smooth read write interconnectedness just like the secure card system.
- Such technologies like FeliCatm and MIFAREtm are based on the protocols according to which the reader is free to access or write data to the card device and then transfer the obtained information through Bluetooth and LAN.

3. Reader/Writer Mode

- The NFC receiver is a reader that read the records of passive RFID devices.

- The RF field is created by the active NFC which acts to power and talk to a passive tag.

RFID

- Radio Frequency identification (RFID) is technology that is utilized in automatic identification of objects based on radio waves.
- RFID systems also utilize net connectivity such as the Internet which can be used to read tags remotely and store and retrieve tag data.
- RFID tag is a label affixed onto a product and this can be used to identify and track the product.
- These tags may be attached on many objects like packages, equipments, animals, birds, and even individuals to track.
- RFID finds extensive application in IoT based business solutions such as in parcel tracking, inventory management, product authentication and monitoring of supply chain.
- RFID is used in the automation of check-in and sales tracing in the warehouse, logistics, and automated warehouse in many industries..
- By attaching an RFID tag to an asset, its movement can be recorded and its status updated in real time through networked systems.
- Bluetooth BR/EDR and Bluetooth Low Energy

Bluetooth Features

1 A device may support Bluetooth Low Energy (BLE) in single-mode, or dual-mode Bluetooth combining BLE with BR/EDR (where Mbps means megabits per second).
2 Bluetooth enables automatic synchronization between smartphones and other devices, using built-in capabilities such as self-discovery, self-configuration, and self-healing.
3 The communication range depends on the radio class used—Class 1 offers about 100 m, Class 2 around 10 m, and Class 3 roughly 1 m.
4 Bluetooth also supports NFC-assisted pairing, which reduces latency and speeds up the device connection process.
5 IoT and M2M networks can employ either dual-mode or single-mode Bluetooth devices within their local area environments.
6 BLE (Bluetooth Smart) provides an option for IPv6 connectivity through the Internet Protocol Support Profile (IPSP).
7 BLE uses smaller-sized data packets to maintain low power consumption and efficient communication.

8 Bluetooth communication can operate in both secure and non-secure modes, allowing devices to choose link-level security, service-level security, or no security.
9 For encryption and data integrity, Bluetooth uses the AES-CCM 128-bit authenticated encryption technique.
10 IoT, M2M, or mobile devices using Bluetooth EDR can connect to the Internet via a 24 Mbps Wi-Fi 802.11 adaptation layer (AMP) or through Bluetooth-enabled wired connection ports.
11 The MAC (Media Access Control) sublayer manages data transmission within the Bluetooth data-link layer.

ZigBee IP / ZigBee SE 2.0

1 ZigBee is based on the IEEE 802.15.4 standard, which defines the physical and data-link layer protocols for WPAN communication.
2 ZigBee devices create a Wireless Personal Area Network used by sensors, actuators, controllers, appliances, and medical-monitoring systems that link to the Internet for various IoT applications.
3 The ZigBee Neighborhood Area Network (NAN) is a version tailored for smart-grid systems.
4 ZigBee Smart Energy 2.0 uses IP networking to support energy management and improved energy efficiency.

ZigBee IP Feature

- The L1 Protocol Data Unit (PDU) in ZigBee has a size of 127 bytes.
- ZigBee IP is intended for low-power, short-range wireless personal area networks.
- A ZigBee device can operate in six roles: end device, ZigBee router, ZigBee coordinator, ZigBee-IP coordinator, ZigBee-IP router, and IP host.
- ZigBee IP adds support for IPv6 connectivity. ZigBee IP devices typically function as Reduced Function Devices (RFDs), meaning they operate in low-power “sleepy” mode and wake only to send or receive data.
- IPv6 is also provided using the compression of 6LoWPAN header which allows the low-power devices to communicate effectively using the Internet.
- Its protocol base is TCP and UDP at the transport layer and base end to end security on TLS 1.2 with RSA, ECC, and PSK cipher suites.
- ZigBee routers support both proactive and reactive routing methods, and thus, they can be adopted in large scale automation and remote-control.
- ZigBee is a self-configuring, self-healing mesh network and records the transmission to unicast communications as well as multicast communications.

- It also has multicast forwarding which is to facilitate mDNS based service discovery..
- The protocol provides mechanisms for device discovery and application-level confirmation.
- Coordinators can pair with routers and end devices in a star configuration.
- Multiple star networks can be combined, allowing creation of larger, interconnected networks with inter-PAN communication.
- ZigBee supports integration of sensor networks and appliances, with nodes configurable as routers or end devices.
- The system supports low-latency communication (<10 ms).
- ZigBee provides a range of 10–200 meters, with a data rate of 250 kbps and low-power operation.
- It operates in the ISM band using 16-channel DSSS radio and ensures link-level security with AES-CCM-128 encryption.
- ZigBee SE 2.0 also incorporates Reduced Function Devices (RFDs).

ZigBee NAN

ZigBee NAN is designed for smart metering, distribution automation, and smart-grid communication.

- It supports last-mile connectivity in utility networks, linking home area networks (HANs) and outdoor access systems to wide-area network (WAN) gateways.

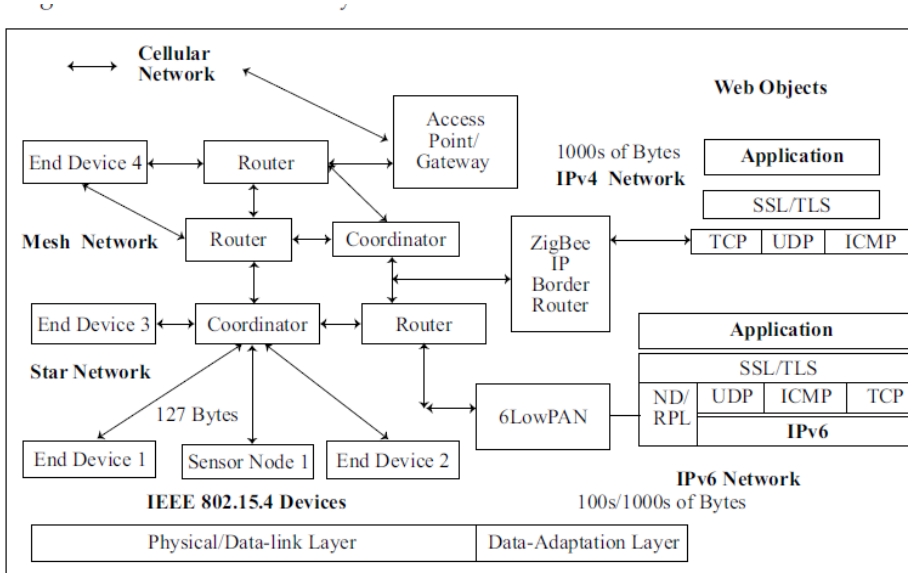


Figure 11ZigBee end point, coordinator, router, ZigBee IP router nodes forming the star, mesh and IP

Three end devices, Two routers, and one sensor node that is linked to coordinator ZigBee. machinery, which is star-shaped. The two routers and one coordinator constitute one end device in a mesh network.

- Mesh network router is connected to an AP/gateway and this is connected to a cellular.
- network.
- Mesh network coordinator is connected to ZigBee IP border router, which allows local.
- Internet connectivity to the ZigBee networks..

ZigBee Network Features

1 In a star-topology ZigBee network, a router connects to a 6LoWPAN layer, enabling IEEE 802.15.4 devices to communicate over an IPv6 network.

2 Thousands of bytes of data can be exchanged between the network layer and IoT web resources or objects.

3 Each individual data transmission at the IEEE 802.15.4 adaptation layer carries 127 bytes, which is the maximum frame size.

4 Protocols such as IETF Neighbor Discovery (ND), ROLL, RPL routing, IPv6/IPv4 stacks, and transport protocols like TCP, UDP, and ICMP—along with SSL/TLS security—support communication between ZigBee nodes and web-based applications or objects.

2.4 Gateway Functions: Data Enrichment, Consolidation, and Device Control

A gateway at the data-adaptation layer has many functions including provision of privacy of data, provision of data security, addition of data, consolidation of information, data format transformation and control of connected equipment. The last level of the model is the device level. The capability in this layer is in the form of device and gateways. The gateway will have two functions i.e. data management and consolidation and attached device management..

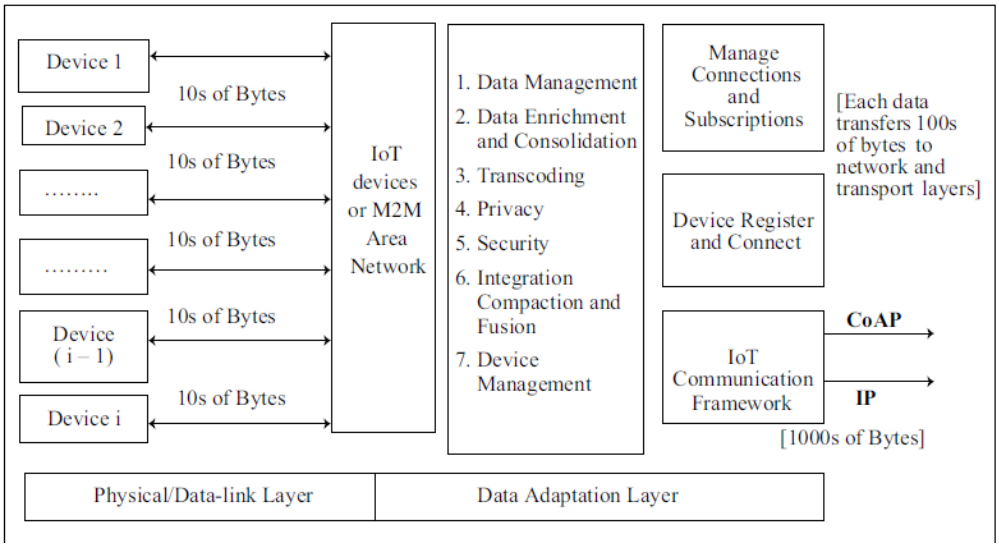


Figure 12Adaptation-Layer Gateway for IoT/M2M: Data Processing, Device Management, and Communication

2.4.1 Gateway for Data Management and Consolidation in IoT Systems

The gateway provides support for multiple functions such as transcoding and data management.

The key data management and consolidation functions at the gateway include:

- Transcoding
- Privacy and security mechanisms
- Integration
- Data compaction and fusion

Transcoding

Transcoding involves transforming data by altering its format, protocol, or coding scheme through software. The gateway formats any web response as demanded by the IoT devices. The transcender, in turn, transforms messages of IoT devices into the formats that the server can understand.

Privacy

Privacy is one of the data management aspects, and it must be taken into consideration when designing an.

application. The design should be able to offer privacy in order to ensure that the data being received is confidential.

end does not have the name of a corporation or a personality.

The aspects of the privacy model are pursued:

- Objectives identity-management Software identity-management
- Authentication
- Authorisation
- Trust

Reputation

It must be ensured in the inter-point transmission of data that the future users of the data do not have access to data on the device level and use it abusively in their application. It has a foundation on trust and reputation in such interactions which are statical and dynamic..

Secure Data Access

The access to the data is critical. It should be designed to confirm the identity of any data request and then it needs to be authorized before any response or any service is availed. It can also include auditing tools that can be used to record requests and responses in case of future accountability.. As shown in Example 2.4, confidentiality and authorization can be achieved using AES-128 with CCM. End-to-end security further strengthens protection by applying security protocols across all layers—physical, link, and transport—on both communicating ends.

Data Gathering and Enrichment: Concepts, Methods, and Applications

IoT and M2M applications perform several core operations, including data acquisition, validation, storage, processing, retention, and analysis. Data gathering refers to the process of collecting data from devices or device networks. There are four primary modes of data acquisition:

1. Polling:

In this mode, the system actively requests data from a device by sending a query.

Example: Checking the fill level of a waste bin in a smart waste management system.

2. Event-based Data Collection:

Data is transmitted only when a specific event or trigger occurs.

Example: A device sending data when it comes near an access point, or when an NFC-based interaction initiates a Bluetooth pairing.

3. Scheduled Interval Gathering:

Devices send data at predefined time intervals. Example: Periodic reports of ambient light levels in a smart street lighting system.

4. Continuous Monitoring:

Data is streamed continuously from the device to observe real-time conditions.

Example: Constant surveillance on a road or continuous surveillance of light on the smart streetlights.

Data enrichment refers to the process of adding value to the gathered information with the aim to make it more valuable, secure, accurate as well as more useful. This can involve metadata addition, data quality validation, data security measures and processing the data in the form of a more useful analysis.

Introduction to the Distributed System of Data Dissemination.

Represented are three main steps of enrichment, namely, aggregation, compaction and fusion.

Aggregation refers to a process of making newly accumulated data frames together with already received data and disregarding any occurring repetitions and redundancies.

The concept of compaction can be described as a procedure of downsizing the amount of data without modifying its meaning or context.

As an illustration, it only includes transmitting changes that are incremental to ensure that the entire text is brief.

Fusion refers to combining information that was gained when different data frames were used or in case there was more than one data source. It eliminates repetitions and structures the integrated components into one comprehensive and consistent data. Data fusion can particularly be applied where the individual records of data are not required or that they cannot be recovered in future.

Energy activities in Data distribution.

Data dissemination consumes a substantial amount of energy, which is a very important aspect of a devices under the number of devices in the WPANs and wireless sensor networks (WSNs), mainly due to the nature of the devices which are usually powered by a low amount of battery. There is energy used when computing and in communication of information. Energy consumption also goes up as the rate of data improves. Equally,

the more the RF transmission power, the more the consumption of energy. On the other hand, the time between the operations of data-gathering can be longer to decrease the consumption of energy in general.

Aggregation, compaction as well as fusion, are techniques which make it possible to handle data efficiently with regard to energy use. Devices can save much on energy dissipation by implementing multi-fold methods such as reducing the number of bytes sent over the air, increasing the data acquisition rate as well as reducing the data transfer rate.

Data Source and Destination

ID:

Each device and any of its internal resources has its own identifier. It ID assists in defining the source of the data (source ID) and destination of the data (destination ID).

Address:

The network header fields consist of destination address in order to deliver data appropriately..

For example, a 48-bit MAC address is used at the link layer, a 32-bit IPv4 address at the IP layer, and a 128-bit IPv6 address in IPv6 networks. In addition, headers may specify application-level ports, such as port 80 for HTTP, to indicate the service or application endpoint.

Data Characteristics and Structural Representation

Data characteristics may be classified into several types, such as:

- Temporal data, which varies with time;
- Spatial data, which depends on geographic location;
- Real-time data, generated and captured continuously at the same rate;
- Real-world data, originating from physical environments (for example, traffic flow, streetlight operations, or ambient conditions);
- Proprietary data, protected under copyright and accessible only to authorized users; and
- Big data, which refers to large-scale and mostly unstructured datasets.

Before being transmitted over the Internet, device-generated data is formatted into specific representations such as XML, JSON, or TLV. Files shared online may also be associated with a MIME type to define their content type.

Data structure refers to the arrangement or organization of data bytes in a defined sequence, usually with a fixed size limitation corresponding to the Protocol Data Unit (PDU) of a particular communication layer.

2.4.2 Device-Management Gateway

Device Management (DM) refers to the processes involved in provisioning a unique device ID or address, activating the device, configuring its parameters and settings, registering and deregistering it, and managing its attachment or detachment from the network.

DM contains the management of resource subscriptions also in which a device accepts or withdraws.

customers subscribe to its services.

The other significant factor that should be considered is fault management which entails the process and the protocol to be used in case a device is faced with an error or malfunction.

A number of standards are used in the management of the devices such as Open Mobile Alliance Device Management (OMA-DM).

A method by which a device-management may be carried out by a gateway performing;

Play the role of a forwarder when both the device and DM server can feature well without reformatting or restructuring messages.

- Conversion of protocols: conversion of protocols can be done in case the device and the DM server use different protocols.
- Can be used as a proxy when intermediate caching or pre-fetching is required, particularly in lossy or unreliable network.
- The OMA-DM model argues that a DM server communicates with devices in IoT and M2M systems via a gateway. Since the DM server manages the process of assigning device IDs or addresses, activating devices, setting parameters and service, it is part of the management of the latter.

subscriptions, and modes of setting devices..

In low-power or high-loss communication environments, a device may communicate via a gateway instead of engaging directly with the DM server.

2.5 Ease of Designing and Affordability

Designing connected devices for IoT applications, services, and business processes focuses on ensuring simplicity at the physical, data-link, adaptation, and gateway layers. This includes having access to SDKs, low-cost prototype boards equipped with smart sensors, actuators, controllers, and IoT modules, as well as hardware that supports open-source software components and standard protocols. Devices should integrate only the essential components and rely on ready-made solutions to facilitate easy development of personal-area networks and secure Internet connectivity.

Ease of design also depends on the availability of convenient features and affordances such as RFID cards or smart cards, which embed microcontrollers, memory, operating systems, NFC interfaces, and RF transceivers at very low cost.

In wireless sensor networks, devices often use Motes, which are inexpensive units equipped with lightweight, open-source operating systems like TinyOS. These Motes are easy to deploy and easier to use in WSN environment.

Devices used in smart homes and smart cities usually use the ZigBee IP, or Bluetooth Low Energy (BLE) 4.2—either single-mode or dual-mode—due to their low cost and simple integration along with their low-energy consumption capabilities.

Some design decisions may however make things harder. Indicatively, take the case of an umbrella that is connected to the Internet and what is needed is to program the umbrella to automatically send out the SMS notifications, and this might need a user manual, which complicates the design of the umbrella. Some of the challenges also related to connected devices are that they need to provide secure communication, as well as ensure that the information is handed to the trusted destination by use of encryption mechanism.

2.6 Summary

The chapter describes the design basics of the IoT-enabled and M2M enabled devices with emphasis being on how the data of the device travels across layered communication models based on the OSI architecture. It explains the important terminology, network structures, and all the functions of the devices including coordinators, routers, as well as end nodes. The standards that are offered by international organizations such as IETF, ITU-T, Etisi, and OGC define the instrumentality of IoT structures and compatibility. In the chapter, there are also highlights of major wireless technologies, the NFC, RFID, Bluetooth/BLE, ZigBee and 6LoWPAN which allow communication in the short range within an IoT system. Gateways are depicted as the key elements in terms of data management, security, enrichment, and device control, which are backed by such

standards as OMA-DM. And lastly, it highlights the point that ease of design of the IoT device relies on affordable hardware, open-source tools, and effective communication technologies, while pointing out that additional functionality and high-security standards may complicate the design of the device.

Deep Science Publishing, 2025
https://doi.org/10.70593/978-93-7185-106-0

DeepScience
Open Access Books

Chapter 3: Web Integration and Connectivity Essentials

3.1 Introduction

These form the basis of developing systems to enable devices to interface effectively across the web, usually in resource-contained contexts (such as IoT devices). Key considerations include:

- Scalability: The ability of the system to handle a growing number of devices and increasing data volume without degradation in performance.
- Interoperability: Ensuring that devices and components from different manufacturers and running different software can communicate effectively.
- Security: Implementing measures like authentication, authorization, and encryption (e.g., DTLS for CoAP, TLS for MQTT) to protect data and devices.
- Resource Efficiency (for IoT): Using lightweight protocols (like CoAP, MQTT) and minimal data formats (like JSON, TLV) to conserve battery life, memory, and bandwidth on constrained devices.
- Reliability: Mechanisms to ensure messages are delivered, even over unreliable networks (e.g., CoAP's Confirmable messages, MQTT's Quality of Service levels).
- Flexibility: Supporting various communication paradigms (e.g., Request-Response, Publish-Subscribe).

3.2 Data Format Standards for Communication

Such require higher standards according to which data is organized to be transferred between devices and web services.

The communication data formats or encoding schemes include JSON, TLV and MIME, which are different and each has its own application as per its structure and performance

attributes. JSON is a relatively lightweight, text based, web API format; TLV is a succinct format more specifically used in networking and embedded systems, and MIME is used in email/internet content type taming various data types.

3.2.1JSON (JavaScript Object Notation)

JSON is a data exchange format that is human readable and written in text that is popularly used in contemporary web development..

Characteristics: It is based on the use of key-value pairs and offers such types of data as strings, numbers, Booleans, arrays, and nested objects. It is syntax-independent but based upon Java-script object syntax..

Example Structure:

JSON{

"temperature": 25.5,

"humidity": 60,

"timestamp": 1678886400

}

Pros:

- Human-readable and easy to debug
- Lightweight and compact compared to formats like XML, resulting in less bandwidth usage.
- Easy to parse and generate by most programming languages.

Cons: Less efficient than binary formats for performance-critical applications due to its text-based nature and metadata overhead.

Use Cases:

- Web APIs (RESTful APIs) for client-server communication
- Configuration files for software applications.
- Data storage in NoSQL databases like MongoDB.

3.2.2 TLV (Tag-Length-Value or Type-Length-Value)

TLV is a binary encoding scheme used for informational elements in communication protocols.

Characteristics: Data is structured into a sequence of triplets:

Tag/Type: A binary code identifying the kind of data.

Length: The size of the value field in bytes.

Value: The actual data, which can be of variable size.

It can also support nested TLV structures.

Pros:

- Efficient in terms of size and parsing speed due to its binary nature
- Extensible and flexible, as new tags can be added without breaking existing systems (older nodes can safely skip unknown tags).
- Easy to parse using simple, generalized functions.

Cons: Not human-readable, making debugging more challenging than with JSON. Requires a predefined understanding of tags between sender and receiver

Use Cases:

- Networking protocols like IP, CoAP, and Bluetooth Low Energy (BLE).
- Smart card applications and payment systems (EMV standard).
- Embedded systems where memory and processing power are limited.

3.2.3 MIME (Multipurpose Internet Mail Extensions)

MIME is a standard for indicating the nature and format of a document or file, primarily for use in email and HTTP to support the transmission of diverse content types beyond simple ASCII text. It is not a data format for structuring data itself in the same way JSON or TLV are, but rather a wrapper or categorization system for the data being sent. MIME defines a set of "media types" (e.g., application/json, image/jpeg, text/plain) to specify the data's format. It handles character sets, multi-part messages, and non-text attachments.

Pros:

- Universal standard for email and web (HTTP/S) communication, widely supported across all platforms
- Allows for the transmission of virtually any file type and complex multi-part messages
- Cons: Adds overhead with headers and encoding, not designed for efficient data structuring within an application logic itself.

Use Cases:

- Email attachments (sending images, documents, etc.).

- HTTP content negotiation (indicating the type of data sent in a web API response or request body).
- Describing data formats in various protocols, including sometimes being used with JSON or TLV formats in LWM2M.

Comparison between JSON,TLV and MIME

Feature	JSON	TLV	MIME
Format Type	Text-based	Binary	Content-type standard/wrapper
Readability	Human-readable	Not human-readable	N/A (describes content)
Efficiency (Size)	Lightweight (text)	Very compact/efficient	Adds header overhead
Primary Use	Web APIs, config files	Networking, embedded systems	Email, HTTP content types
Flexibility	Flexible schema	Extensible tags (backward compatible)	Supports diverse content types
Parsing Speed	Fast for text	Very fast (binary)	N/A (parser depends on content type)

3.3 Constrained Application Protocol (CoAP)

CoAP is a web transfer protocol formulated especially to be used in constrained networks and constrained nodes (small machines).

- Communication Model Request/Response, like HTTP, but in an extremely minimal binary format. It is implemented on a RESTful architecture and employs the methods of the GET, POST, PUT, and DELETE.
- The transport layer - UDP (User Datagram Protocol). This is one of its drawbacks, whereby UDP is connectionless and does not need the elaborate hand shake found in TCP, and it is therefore more fast and power saving.

Key Features:

- Low Overhead: It has a small fixed header of 4-bytes.
- Optional Reliability: Oliver offers a very simple way to achieve reliability by means of Confirmable (CON) messages out of which an Acknowledgement (ACK) is requested.
- Multicast: Has the option to use the multicast feature in UDP to do one-to-many communication which would be handy in management of devices locally.

CoAP Extensions (CoAP-SMS and CoAP-MQ)

.

Protocol Extension	Description	Use Case
CoAP-SMS	Transmits CoAP messages over SMS (Short Message Service).	Used in scenarios where a device loses IP connectivity (Wi-Fi/Cellular Data) but can still communicate over the standard GSM SMS channel. Provides a crucial low-bandwidth fallback.
CoAP-MQ	A mechanism to bridge CoAP networks with Message Queue (MQ) systems, often an MQTT broker.	Enables constrained CoAP devices in a local network to publish data to, or receive commands from, a broader, cloud-based MQTT system via a gateway that translates the protocols.

3.4 Message Queuing Telemetry Transport (MQTT)

MQTT is an extremely lightweight messaging protocol for resource-constrained devices, focusing on reliable data delivery over unstable networks.

Communication Model: Publish/Subscribe. Clients (devices) publish messages to a named Topic, and a central entity called the Broker distributes those messages to all interested Subscribers.

Transport Layer: TCP (Transmission Control Protocol). This provides a reliable, ordered, and error-checked stream, which is why MQTT doesn't have to build its own complex reliability layer.

Key Features:

- Broker-Centric: Decouples publishers from subscribers, improving scalability and reliability.
- Quality of Service (QoS): Offers three levels of delivery assurance (QoS 0: At most once, QoS 1: At least once, QoS 2: Exactly once).
- Session Management: Supports persistent sessions and "Last Will and Testament" (LWT) messages to notify subscribers if a client disconnects unexpectedly.

3.5 Extensible Messaging and Presence Protocol (XMPP)

XMPP is an older, more feature-rich, and complex protocol that has been adopted for IoT, primarily for its focus on real-time presence and federation.

Communication Model: Originally designed for Instant Messaging (IM), it uses an asynchronous, XML-streaming approach. It supports both Client-Server and an optional Publish/Subscribe (PubSub) extension.

Transport Layer: TCP. It maintains persistent connections for real-time presence.

Key Features:

- XML-Based: Data is sent as XML "stanzas" (structured units), which makes it highly extensible but also heavier than the binary formats of CoAP and MQTT.
- Presence: Excellent for applications that need to know the real-time availability (online, offline, busy) of devices or users.
- Decentralized/Federated: Supports server-to-server communication, allowing multiple separate XMPP domains to communicate (similar to email)

Difference between COAP and MQTT protocols

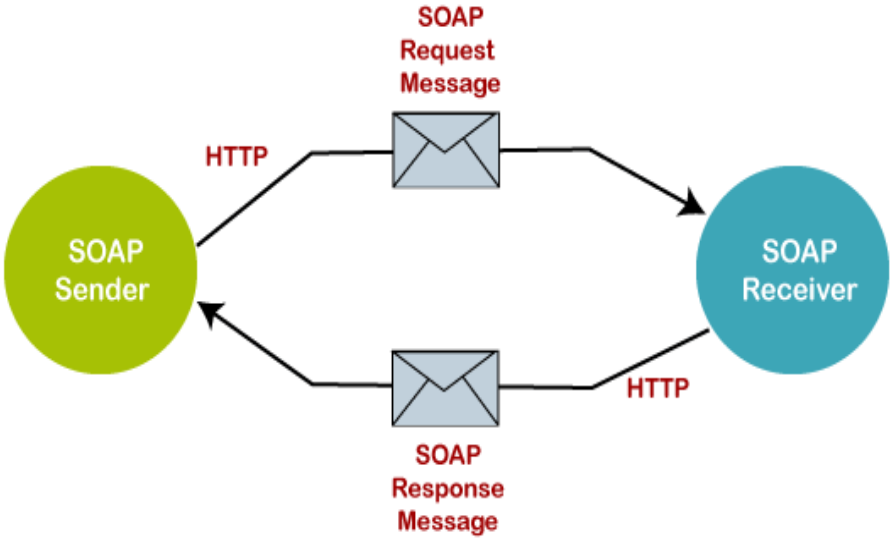
Basis of	**COAP**	**MQTT**
Abbreviation	Constrained Application Protocol	Message Queuing Telemetry Transport
Communication Type	It uses Request-Response model.	It uses Publish-Subscribe model
Messaging Mode	This uses both Asynchronous and Synchronous.	This uses only Asynchronous
Transport layer protocol	This mainly uses User Datagram protocol(UDP)	This mainly uses Transmission Control protocol(TCP)
Header size	It has 4 bytes sized header	It has 2 bytes sized header
RESTful based	Yes it uses REST principles	No it does not uses REST principles
Persistence support	It does not has such support	It supports and best used for live data communication
Message Labelling	It provides by adding labels to the messages.	It has no such feature.
Usability/Security	It is used in Utility area networks and has secured mechanism.	It is used in IoT applications and is secure
Effectiveness	Effectiveness in LNN is excellent.	Effectiveness in LNN is low.
Communication Model	Communication model is one-one.	Communication model is many-many.

3.6 SOAP (Simple Object Access Protocol)

SOAP is a formal, XML-based protocol for exchanging structured information in the implementation of web services. The gateway deploys SOAP for environments where strict contracts and high reliability are mandatory.

Communication Model: Procedure-call oriented, based on a request/response pattern.

Data Format: XML only. Messages are structured using a formal XML Envelope that contains a Header (metadata) and a Body (payload).



Gateway Role:

Enterprise Integration: Enterprise integration is utilized to integrate with legacy system or enterprise systems (such as banking or healthcare) that require formal standards and adherence.

Security: Imposes WS-Security, which can be configured to enable encryptions of messages, digital signatures and high authentication into the gateway layer.

Reliability: The gateway can manage the delivery of messages over unreliable networks using such standards as WS-ReliableMessaging to ensure that messages are delivered.

REST (Representational State Transfer) is a protocol addressed during the 7th phase in the architecture design process.

3.7 REST (Representational State Transfer) REST (Representational State Transfer) is a protocol discussed in the 7th phase of the architecture design process.

The architectural style of distributed hypermedia systems is called REST. The gateway is constructed on the concepts of REST to develop resource-based interface that is scalable.

Model of Communication Client-server, stateless, cacheable. It establishes a guideline in which one can design web services.

One of the /Main ideas/Curisors Everything is handled as a resource (e.g. a device, a sensor reading, a user) which has a specific identifier (URI) which is unique.

Gateway Role: This is where the gateway serves as the one prominent entry point to the underlying network of devices to reveal their capabilities in the form of abstract resources, the logical form achievable over the REST constraints.

3.8 HTTP RESTful

RESTful in particular is used to describe an API that is based on the REST architectural style and implemented on the HTTP protocol. It is the most popular and commonly used strategy, used by the communication gateways.

methods/verbs: It directly translates CRUD (Create, Read, Update, Delete) operations to the normal HTTP methods:

GET: Read/Retrieve resource state (e.g. GET / devices/123/ status).

POST: issue a new resource or perform a non-idempotent action (e.g. POST / devices/ logs).

PUT/PATCH: Grenzlike an already existing resource (e.g. PUT /devices/123 to change state to new one).

DELETE: Remove a resource.

Data Format: Flexible, most often, JSON (JavaScript Object Notation), however, may also be XML or plain text.

Translation of protocols: The most important of functions. It converts incoming HTTP RESTful messages into the native IoT protocols (such as MQTT or CoAP) and converts the response of the device to a single regular HTTP format (such as a JSON payload, etc.).

Stateless: The transaction of every HTTP request is processed separately, and can hence be scaled horizontally (additional instances).

Caching: Leverages the in-built HTTP caching solutions (such as ETAG) and cache control headers to save commonly used configuration of the device thereby minimizing the load borne by devices.

3.9 WebSockets

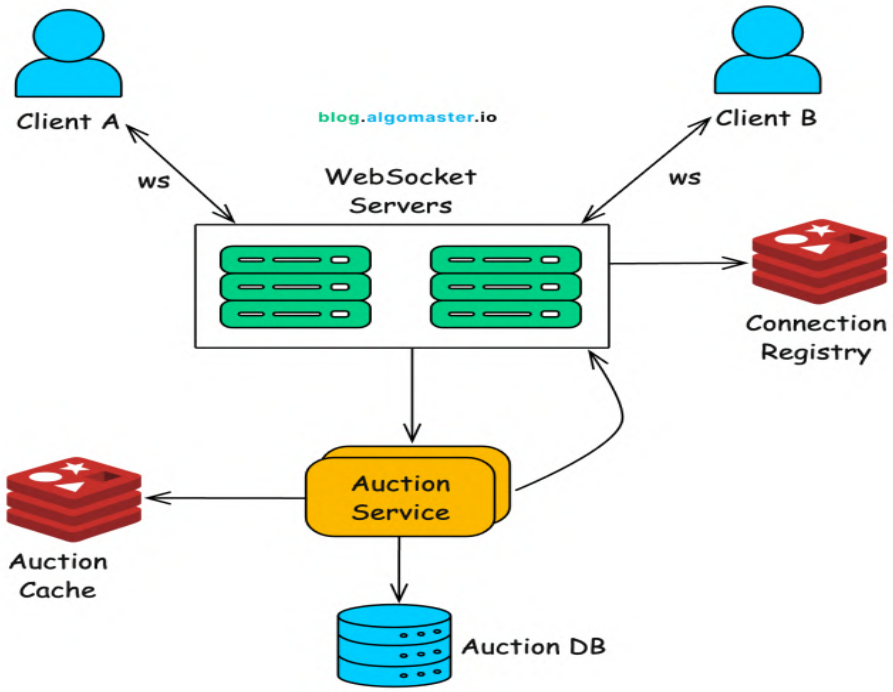
WebSockets is a unique protocol; it offers fully-duplex (two-way) communication channels based on a single long lived TCP connection.

Communication Model: Two way, event-based, continuous communication back and forth. Once the connection has been established, it turns to be stateful.

Transport: An HTTP handshake is used (an Upgrade request) and it switches connection to the specific HTTP protocol of Web OS..

Key Features:

- Real-time: Allows the server (gateway) to push data to the client (app) without the client needing to constantly request it (polling).
- Low Latency: Eliminates the overhead of establishing a new connection and sending full HTTP headers for every message.



Gateway Role:

- Real-time Data Stream: his is applicable as a live sensor data feed or live alerts, chat and stock-ticker styled data feeds.
- Persistent Control: mages 10-25 Intermittence persistence is necessary to manage the network; the constant delay between actions and messages must stay low, enabling a subordinate connection to proceed without any accident.
- Unrelenting Control: offers a dependable, low-latency trigger to transfer commands to any system which demands immediate response. The constant interval between activities and messages cannot be permitted to exceed any limit in order in order to allow a sub-link in the connection to continue without mishap.
- Bi-directional Bridge: This also detects real-time events directly on underlying device Pub/Sub protocols (such as MQTT), and transmits them, and the opposite, to the persistent connection in WebSocket frames..

The communication gateway deploys this mix of methods to satisfy different client needs:

Method	**Primary Communication Pattern**	**Key Use Case**	**Advantage**
SOAP	Procedure Call (Request/Response)	Enterprise, high-security, transactional systems.	Strong security (WS-Security), high reliability (WS-ReliableMessaging), formal contracts (WSDL).
HTTP RESTful	Resource-based (Request/Response)	General web/mobile application data retrieval and simple commands.	Simplicity, wide adoption, excellent scaling and caching via HTTP.

3.10 Summary

This chapter defines the most important web connectivity concepts and communication protocol applied to the IoT systems with a focus on scalability, interoperability, security, reliability, and performance mentioning efficient exchange of data flow. It describes some of the most important data formats like JSON, TLV and MIME, and their use in the organization and discussion of information between devices and services. Other

protocols of the IoT communication vital communication protocols, such as CoAP to handle lightweight RESTful operations using UDP, MQTT to provide publish/subscribe services over TCP, and XMPP to handle XML based real time communication, are also described in the chapter and their features compared. Moreover, it addresses enterprise-oriented SOAP services, the principles of REST architecture, the client-server communication via the use of HTTP RESTful API, and bi-directional real-time data streaming via WebSockets. In general, the chapter is a lucid vision of the manner in which gateways combine these protocols to offer uninterrupted, safe, and effective communication through IoTs..

Deep Science Publishing, 2025
https://doi.org/10.70593/978-93-7185-106-0

DeepScience
Open Access Books

Chapter 4: Foundations of Internet-Based Connectivity Systems

4.1 Introduction

The Internet is a global network that uses a set of connectivity protocols to accomplish the following:

- Provide gateways for connected devices to send and receive data frames. Data travels across the network in the form of packets, which are forwarded through multiple routers on the Internet. Various processes manage, collect, organise, and analyse IoT device data for applications, services, and business operations.
- Enable devices to perform monitoring and control functions using messages, data stacks, and commands delivered through the Internet by applications, services, or business systems.

The following key terms are essential for understanding the principles of Internet connectivity between networks of connected devices and IoT applications:

- **Header**: A collection of 32-bit words that contain control information required for processing a data stack at a layer. Each header wraps (encapsulates) the data from the layer above before passing it to the layer below. Each word may contain one or more header fields, defined according to the processing required at each stage up to the destination.
- **IP Header**: A set of header fields consisting of parameters encoded according to the Internet Protocol (IP). IP operates at the Internet layer at both the source and the destination.
- **TCP Header**: A set of header fields with parameters encoded according to the Transmission Control Protocol (TCP). TCP operates at the transport layer at the source and destination.
- **Protocol Data Unit (PDU**): The maximum-sized data stack that can be processed by a specific layer or sublayer in accordance with its protocol.
- **TCP Stream (rewritten from "alter sentence**"):

A TCP stream is a continuous sequence of bytes generated at the transport layer and transmitted reliably to the transport layer at the destination.

Maximum Transferable Unit (MTU) This is the largest size of a transmission that an upper protocol layer can ever send to a lower layer or to a physical network.

A packet consists of a series of bytes of protocol-specificably defined maximum length that is sent by the network layer but passes through routers to the physical, data-link and network layers on the receiver side. Internet technologies offer packetisation mechanisms of the data stack at the Internet layer and passes it to the following lower layer. The Internet layer at the receiver reassembles (presents packets to the transport layer) the packet at the receiver. An IP packet refers to a piece of data consisting of an IP header and is sent through routers between a source IP address to a destination IP address via the help of the interactive routers.

Data segment is the one that is supplied to the transport layer by the layer of application support. Application data may be larger than the size of the used transport-layer, in which case it is subdivided into several segments. A network interface is a software or a hardware parts that facilitate the interaction of protocol layers, computers or nodes within a network. The interface software offers unified services like connection setup, connection breakdown, and relaying of messages. Some examples of network interfaces are ports (software or hardware), network interface hardware, and sockets. There is special addressing done to an interface in terms of a port number, socket ID, or node ID. A network interface is a port that is characterized by a protocol whereby the application-layer data are sent to the lower layers to be transmitted and the data coming out of the lower layers in the destination is received by a port known as a port. There are protocol-specific numbers allocated to every port. As an example, the application-layer protocol of HTTP is given Port 80.

A socket is a software interface that consists of integrating both an IP address and a port number, in which applications are capable of receiving and transmitting data. Internet communication may be considered to be between sockets of host databases of the sender and receiver.. A host is a device or node connected to a computer network that provides information, resources, services, or applications to other nodes. The network layer assigns each host a unique address. An IP host is a host that uses the Internet Protocol suite and has one or more IP addresses assigned to its network interfaces. A subnet (subnetwork) is a logical subdivision of an IP network. Subnetting enables a group of networked computers to share a common network prefix. For example, an organization with 1024 computers may assign them IP addresses sharing the same most significant bits (MSBs). The IPv4 address is 32 bits in length and has two sections; the network/ MSBs and the host/ LSBs.

Routing Prefix: Routing Prefix IP address may be subdivided into most significant bits (MSBs) of 8, 16 or 24 bits and the least significant bits (LSBs). This division with reference to the IP address logically divides the address into two fields, which are network address or routing prefix and rest field (host identifier). The rest field is only used to distinguish one of the host/network interfaces..

Host Identifier: The rest field can also be further divided into two components one subnet ID and other one host identifier also which is used when a network is subdivided to subnets with more than one host each..

Data Flow Graph (DFG): DFG is a visual diagram where circles illustrate the processing steps whereas the arrows show the data flow direction.. Inputs (e.g., routing information) at each stage are processed to produce outputs. These outputs flow to the next stage, continuing sequentially until the final output is produced. Thus, data flows from the initial stage to the final stage, being processed at multiple intermediate stages.

Acyclic Data Flow Graph (ADFG): An ADFG is a type of DFG in which a given set of inputs produces a single, unique set of outputs, without forming any loops. All inputs in an ADFG are assumed to be available simultaneously (no delay), except for the processing time at each stage. Examples of non-acyclic inputs include event-driven inputs, device status flags (set = 1, reset = 0), and inputs dependent on the output of a previous process.

4.2 Internet

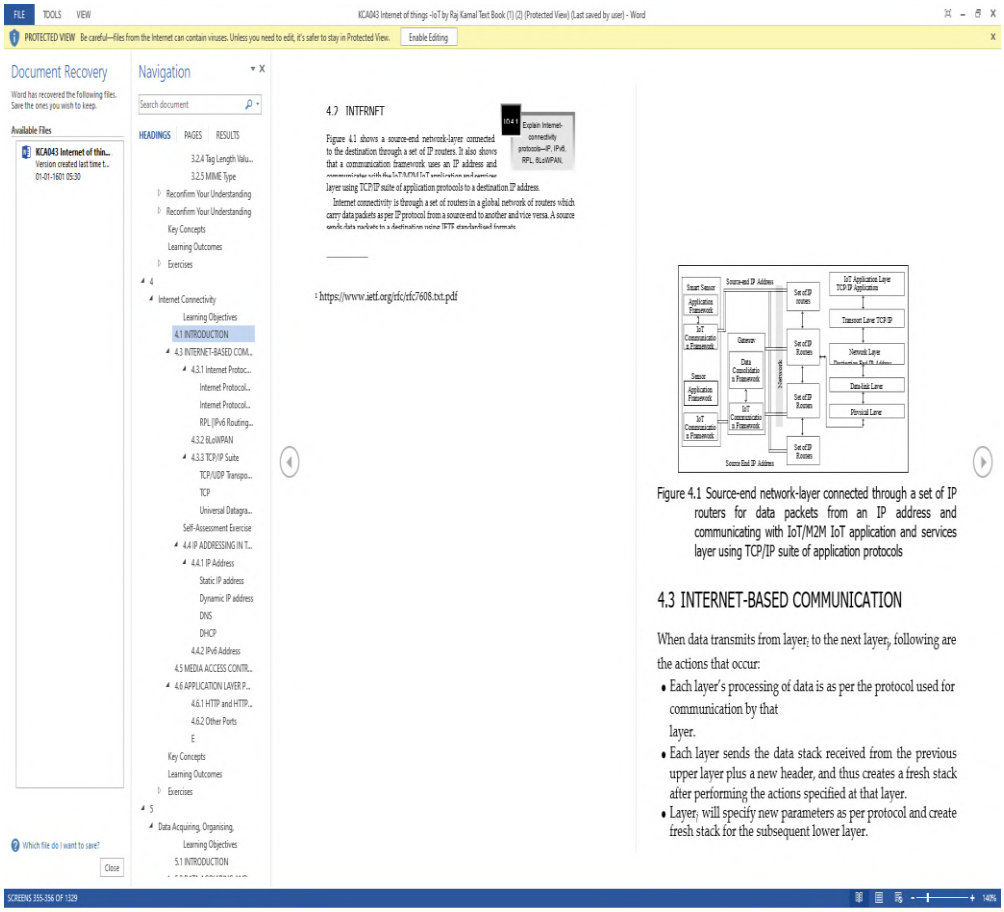


Figure 4.1 illustrates a source-end network layer connected to the destination through a series of IP routers. It also shows how the communication framework uses an IP address to interact with IoT/M2M applications and services.

4.3 Internet-Based Communication

When data is transmitted from layer i to the next lower layer j, the following actions take place:

- Each layer processes the data according to the communication protocol defined for that layer.
- Each layer appends its own header to the data stack received from the upper layer, forming a new stack after carrying out the required operations.

- Layer j sets new protocol parameters and generates a fresh stack to be passed to the next lower layer.
- This sequence continues until the data has been completely transmitted across the network.
- Keep in mind that in the modified OSI model, the IoT application layer is the highest layer and the physical layer is the lowest (Figure 2.1). Therefore, data flows from layer i to layer j (where i > j) when moving from the application layer down to the physical layer.
- When data is received at layer i from a lower layer j—for example, from the IoT device's physical layer up to the IoT application layer—the following actions occur:
- Each layer processes the data based on the header field bits it receives, interpreting them according to the protocol in order to decode the required information and perform the necessary actions.
-

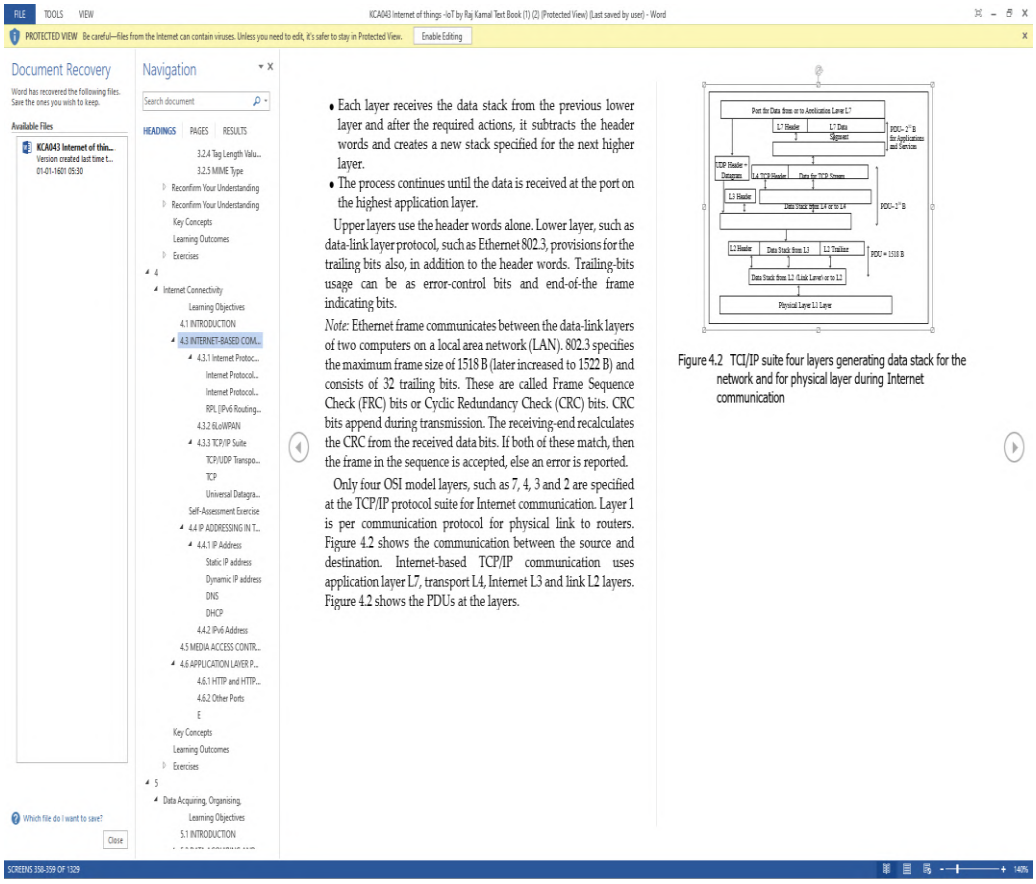


Figure 4.2 TCI/IP suite four layers generating data stack for the network and for physical layer during Internet communication

Figure 4.2 illustrates how the four layers of the TCP/IP suite generate the data stack for both the network layer and the physical layer during Internet communication.

4.3.1 Internet Protocols

The Internet layer receives data from the upper layer and forwards it to the next layer using either Internet Protocol version 4 (IPv4) or Internet Protocol version 6 (IPv6).

Internet Protocol Version 4

Assume that the number of header words is denoted by n. The data stack that the network layer receives or transmits is shown in figure 4.3. The amount of words in an IP packet is n of header fields (160 bits of mandatory header with optional extension words). The ## suffixes are inserted upon the need to take further actions depending on data stack which is inputted by or outputted to the transport layer.

The Internet layer protocol, which is also known as IP, is the one that is used when data transmission is not acknowledged. The data that is received by the Internet layer in transfer between the transport layer and receiver end is sent over a fragment of IP packets. One IP packet comprises the protocol data unit at this layer, PDUIPIP IP and has a maximum size of 216 bytes. This is the highest possible data packet that can be transferred or received at the Internet level with the help of IP packets.

Figure 4.3 shows the structure of an IP data stack, including the IP header fields. The figure indicates the starting and ending bit positions of each header word and the data portion. The IP packet contains a fixed 160-bit header and, when necessary, an extended header of (n–5)(n – 5)(n–5) additional words, along with the data stack of len words received from or destined for the transport layer.

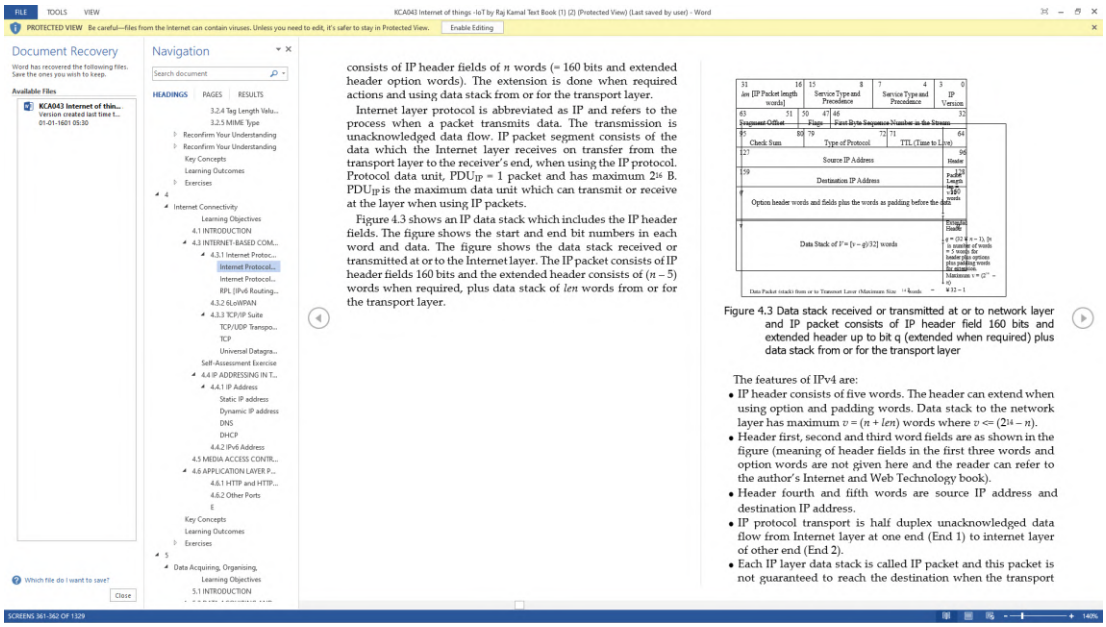


Figure 4.3 shows the data stack received from or transmitted to the network layer. The IP packet includes a 160-bit IP header and an extended header up to bit q (added when required), along with the data stack from or intended for the transport layer.

The key features of IPv4 are as follows:

- The header of the IP has five fixed words and in case of necessity, option and padding words can be added. The data stack that is sent to the network layer has up to v = (n + len) words where v [?] (214 - n).
- Fields of the first, second and the third words of the header are indicated in the figure. These fields (as well as the optional header fields) are not described here, but a reader can see the descriptions of their fields in the Internet and Web Technology book by the author.
- The fourth and fifth header words have the source IP address and the destination IP address respectively.
- Commonly, IP protocol communication is a half-duplex type and is an unacknowledged flow of data between the Internet layer in one end (End 1) at one end, and the Internet layer in the other end (End 2).
- The data stack in each Internet-layer is known as IP packet. The transport-layer protocol can be either UDP or TCP, meaning that the delivery of IP packet is not guaranteed in the former, but guaranteed in the latter.
- A packet informs unidirectional communication at a particular time.

Internet Protocol Version 6

- The IPv6 version has the following features:
- Offers a much increased addressing space.
- Allows the hierarchical addressing and permits the route aggregation throughout the Internet which lowers the routing table growth.
- Plays better optimisation towards service delivery with router and sub nets as well as interfaces.
- Manages device movement, security and setup.
- Increasesand streamlines the multicast addressing.
- Sponsors jumbo grams (large-size datagrams).
- Enables extension with optional headers.

The IPv4 address has a length of 32 bits whereas IPv6 address has a length of 128 bits. The IPv6 therefore provides a significantly large address space as compared to IPv4. IPv6 address is a number name that is used as an identifier of a network interface on a node, and other nodes and subnets on an IPv6-based Internet. One device which is a node uses this address to communicate with the network.

RPL [IPv6 routing protocol of Low-Power and loss networks (LLNs)].

A Low-Power and Lossy Network (LLN) is a type of a network that consists of nodes that have limited power level, the low data transfer rate as compared to conventional IP networks, limited rate of packet delivery, and intermittent or unreliable connection. IETF has stipulated the RPL routing specifications when such ROLL (Routing Over Low-Power and Lossy Networks) environment routing is used. RPL is used in a non-storing routing mode..

In IoT/M2M low-power and lossy environments, the RPL protocol is used. When IEEE 802.15.4 WPAN devices are involved (as shown in Figure 2.5), the IPv6 Internet layer receives and transmits data to and from the data adaptation layer (Figure 2.1).

Low-power nodes must limit communication to the nearest, or at most the next, level either upward or downward in the network. In a lossy environment, the use of disjoint nodes is essential so that, in the event of an error or missing acknowledgment, retransmission can occur through the previously used path. Data is transmitted in an optimally sized unit at any given instance.

Node-to-node data flow follows the Destination-Oriented Directed Acyclic Graph (DODAG) model. A Directed Acyclic Graph represents the direction of data flow among nodes. “Destination-oriented” indicates that communication is structured in a tree-like manner—upward toward the root for transport, or downward toward the device-layer end nodes. The following example further explains the DODAG-based data flow approach used in RPL.

a. Provide an example of the DODAG data-flow approach for RPL, assuming routing nodes arranged across four ranks (levels) 0, 1, 2, and 3, consisting of 3, 6, 3, and 6 nodes respectively. Assume that the node communicates within an RPL-based network environment.

b. Explain how paths—referred to as RPL instances—are established for data flow in a DODAG.

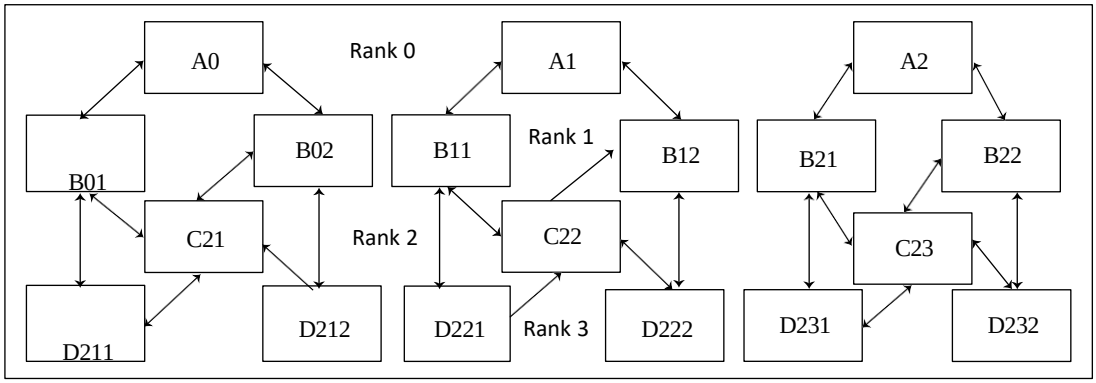
c. List the characteristics of a DODAG using the nodes described in the example.

Solution

(a) An access point can serve as the root of the tree that enables DODAG-based data flow in an RPL network. Consider the following arrangement:

- A0 is the root node at rank 0, with A1 and A2 as additional nodes at the same rank.
- B01 and B02 are child nodes of A0 at rank 1. They maintain upward connectivity to rank 0 and downward connectivity to nodes at rank 2.

- C21, C22, and C23 are the three nodes at rank 2. Each of these nodes has upward connectivity to the B-nodes at rank 1 and downward connectivity to nodes at rank 3.
- D211 and D212 are the child nodes of C21 at rank 3; D221 and D222 are child nodes of C22; and D231 and D232 are child nodes of C23.



The figure 4.4 shows the RPL network that has the nodes spread over four ranks and the RPL data-flow instances of the upward and downward communication.

Rank 0, 1, 2, and 3 are thus occupied with 1, 2, 3, and 6 nodes respectively which take part in the communication in the network. Suppose the devices and destination access point are ranked 3 and 0 respectively. The DODAG structure in RPL provides that all upwards data movement is between a rank-3 node and a rank-2 node, between a rank-2 node and a rank-1 node as well as between a rank-1 node and the destination node in rank 0. There can also be upward data flow in a direct rank-3 node to a rank-1 node in the case that there is enough power available at node D211.

4.3.1 6LoWPAN

The adaptation layer interchanges data with IPv6 Internet layer (Figure 4.1). At the adaptation layer the data stack employs the 6LoWPAN (IPv6 over Low-Power Wireless Personal Area Network) protocol and then is sent on to the IPv6 Internet layer. A device in the IEEE 802.15.4 WPAN uses 6LoWPAN interface in a serial port to connect.

6LoWPAN is a protocol that implements the adaptation-layer to the IEEE 802.15.4 network devices, which by default have a low speed and low power usage. The devices are used as the nodes of a multi-device mesh network (WPAN). Since low-powered nodes have to limit the size of transmitted data transmitted, data compression and fragmentation techniques have been employed to reduce the size of the packets. The

main characteristics of 6LoWPAN are compressions, fragmentations and reassembling of the header. In cases where data is broken to be transferred, a header that has datagram size (11 bits) in it and datagram tag (16 bits) determine the fragment that is sent out first. Following fragment has an 8-bit header that contains datagram size, datagram tag and fragment offset. Fragments can have reassembly time that is set to 60 seconds.

Figure 4.5(a) shows the physical layer of an IEEE 802.15.4 WPAN of networked devices; Figure 4.5(b) shows the data-link sublayer and web protocol, 6LoWPAN adaptation-layer protocol.

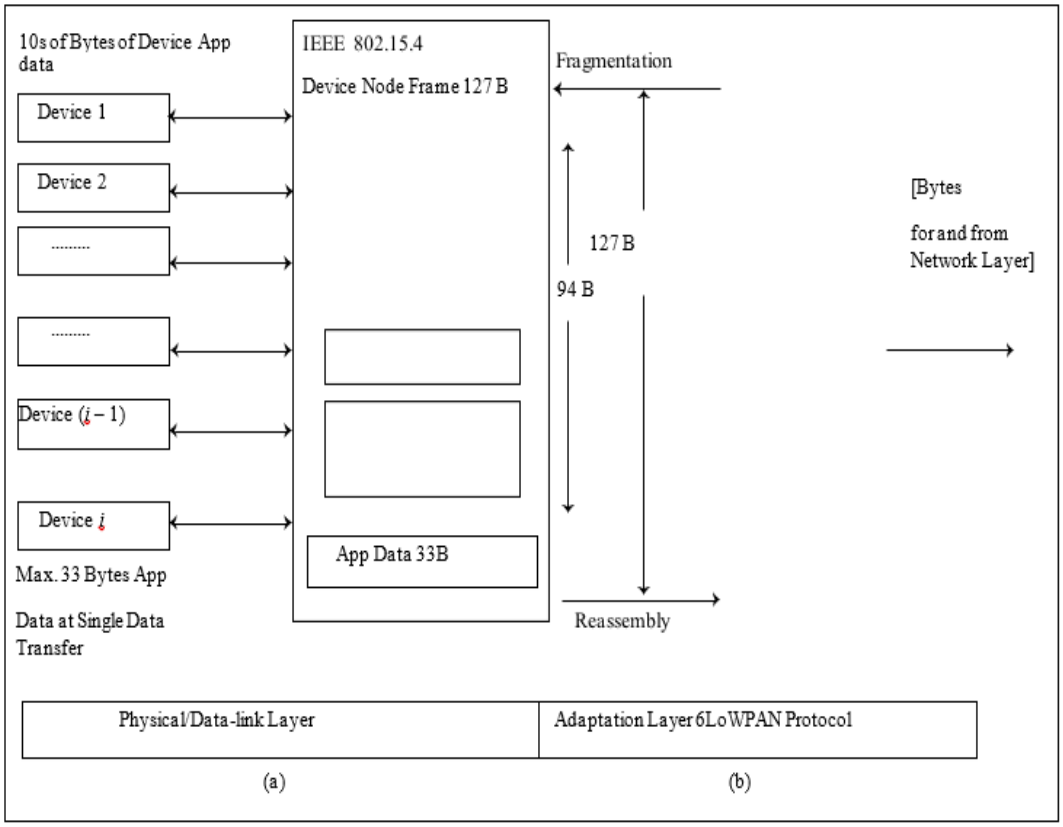


Figure 4.5 shows (a) networked devices at the physical layer in an IEEE 802.15.4 WPAN and (b) the 6LoWPAN adaptation layer, where 127-byte (maximum) fragmented frames are reassembled into an IPv6 packet of up to 1280 bytes, or conversely, an IPv6 MTU of 1280 bytes is fragmented into 127-byte frames for transmission to a device.

Figure 4.5 illustrates how IPv6 operates over an IEEE 802.15.4 network, showing the use of headers, security fields, and application data. The header sizes are as follows: IPv6

header — 40 B; UDP header — 8 B; device node MAC address — 25 B; AES-128 security — 21 B. Since the maximum frame size for a device node is 127 B, only 33 octets remain available for application data. The MAC sublayer functions at the data-link layer (see Example 2.4 and Section 2.4.1 for details on AES).

Because the IPv6 Maximum Transmission Unit (MTU) at the link layer is 1280 B, fragmentation is required to transmit data over IEEE 802.15.4 nodes, which support only 127-byte frames. Thus, multiple 127-byte fragments from the device are reassembled into a full IPv6 frame. Likewise, an IPv6 MTU of 1280 B is fragmented into 127-byte frames for transmission to a device node.

The 6LoWPAN adaptation layer provides the following capabilities:

● IETF-recommended mechanisms for fragment reassembly, header compression for IPv6 and UDP (or ICMP) via the 6LoWPAN-hc layer, and neighbour discovery through the 6LoWPAN-ndlayer.

4.4 IP address

An IP header contains the source and destination IP addresses. While the Internet traditionally uses IPv4 addressing, IoT/M2M systems primarily rely on IPv6. The following subsections explain these addressing methods in detail.

4.4.1IP Address

The IP version 4 address is made up of 32-bits. But, it could be viewed in the form of four decimal figures with dots. For example, 198.136.56.2 for 32 bits--11000110 10001000 00111000 00000010. Every octet number represents a decimal value of a decimal number (= 8 bits). The IP addresses may be either 0.0.0.0 to 255.255.255.255; the total being 232 addresses in all since it is the 32-bit address. The division into three separate fields, one of which should have a decimal number in each group of 8 bits, is easier to work with. We shall have an analogy of postal network addressing method. Consider an address:

The IP address 198.136.56.2 has public Internet appearance. Number of servers, including web server, mail server and FTP server is transparent and shares the same IP address throughout the world and different addresses within a subnet within a server or nodes or devices.

Use of IP address is to identify a single network interface of a host. It is the address which the interface uses to locate itself on the network. The speech allows the packets of IP to be routed among the hosts. The fields of packet header contain IP addresses to do routing. The packet header shows source as well as the destination of the packet.3

Static IP address

One of the IP addresses is the one assigned by the Internet service provider and it is known as a static IP address. The service provider can give one address to an individual. When a single company contains multiple hosts, a service provider can assign a class C network address made up of group of 254(= 28) how the assignments of fixed addresses are made to a cluster of hosts.

Nowadays a new scheme is utilized known as Classless Inter-Domain Routing (CIDR). As an illustration, the Class C 198.136.56.0/4 denotes that the Class C allocated four Internet routing IP addresses to the 4 public domain servers. Each of the 4 IP addresses is assigned the server by the system administrator of a firm with the same domain name.

Dynamic IP address

Having connected with the Internet, a device must be assigned its own IP address. When the device goes connected to a router, the router and the device use the DHCP (Dynamic Host Control Protocol) whereby an IP address is assigned to the device at an instance. This speech is referred to as dynamic ip address. In the event that a device is switched off or disconnection occurs along with the re-booting of the router, the dynamic IP address is lost, and a new one is allocated as the device becomes connected once again.

DNS

Consider an IP address, 198.136.56.2 (11000110 10001000 00111000 00000010). It is not easy to memorize or recall. The address is registered with domain name rajkamal.org.The domain name also allows access to web-server within the domain, as the name of the site, which is, http://www.rajkamal.org/. The domain is used to get access to mail-server hosted in domain by using mail server name which is, http:// mail.rajkamal.org/. The other instance of domain name is mheducation.com/.

Top Level Domain (TLD) are domain com, dom org, dom in and dom us. LD could also be further subdivided in terms of either .co.in or .gov.in. or .gov.uk.

A domain name is offered at a fixed fee annually by a registrar. DNS server at the Registrar contains a control panel (cPanel) and could be set up using cPanel. Domain Names System (DNS) is an application that gives an IP address of a corresponding service in the named domain service.

DHCP

When a sensor or actuator or an IoT device or node wants to make contact with the Internet, a server is referred to as DHCP server that supplies dynamic IP address, subnet mask, ARP and RARP caches. The subnet (server) has its own IP address to allow accessibility to the Internet.4

Dynamic Host Configuration Protocol (DHCP) is a protocol to offer new IP addresses and configure subnet mask dynamically to the connected node to enable it use the subnet server and subnet router at the communication framework.

IP addresses are configured. This is done by the administrator or user. DHCP allows this to be configured automatically during start-up of IP address allocations.

- There is a software element of requesting the DHCP server among a node and receiving a response. The component is known as DHCP client. DHCP client protocol is in communication with a server. The DHCP protocol involves steps to dynamically configure the IP address among others, the steps include:
- The DHCP client sends a discover-request, called DHCPDISCOVER.
- DHCPDISCOVER is received by DHCPserver and locates the configuration which may be provided to the client. The server(s) broadcasts the configuration parameters, which includes an IP address which is not currently in use, at the subnet. The offered configuration has DHCP OFFER configuration parameters.
- The given DHCP server generates and maintains bindings. DHCP server also sets a time
- period associated with the current IP address that the DHCP client node shall have.
- The binding is checked by DHCP server by using a message. It sends DHCPACK out of the creation of the binding.
- In case the node containing the DHCP client computer is no longer in the subnet, it broadcasts a DHCPRELEASE. In case the client fails to send DHCPRELEASE in specified time-period, a created binding is freed by the server.
- Authentication protocols are also undertaken on the server and client direction before taking into account the.

Respectively, DHCPDISCOVER by a client and accepting DHCPOFFER.

The DHCP protocol will ensure that any assigned network address, at a particular time, is occupied by a single, or none, DHCP client.

4.3.1 IPv6 Address

IoT requires high number of addresses per device (nodes). IPv6 uses 128-bit address. The hexadecimal number signifies 4-bit, 0 hex means 0000 binary to f hex means 1111 binary. As such, 128-bit address has 32 hexadecimal words. An IPv6 address consists of 8 sets of 4 hex-digits, which are separated with a colon or a dot. E.g. 16-hexadecimal numbers, 40a0: 0acb:8a00:b372:0000: 0000:0000:0000. IANA is the organization that

takes care of how the IPv6 addresses are allocated, the final part has 64 bits, the last of which can be left out when all the preceding 64 bits are zero.

Mesh network devices can be configured to use 6LoWPAN protocol at adaptation layer and Ipv6 when the communication frameworks are configured to communicate using IPv6 on the Internet. Final 64 bits are identifiers of interface. A commonplace interface can possess a separate node.

The IPv6 addresses are categorized under three classes. The classes vary in the primary addressing and routing. Interfaces can occur at a separate node. A single network address for unicast address is specified by 48 bits or more routing prefi/. 16 bits or less routing subnet id. 64 bits interface identifiers.

Anycast address refers to group of address of nodes or interfaces. A packet has been relayed to an anycast address to only one of the member interfaces. One may be the nearest host. Nearest is calculated based on definition of distance by the routing protocol.

Multicast address refers to the address that is shared by many hosts and it assigns the destination of multicast address through interaction of multicast distribution protocol with network routers. The transmission of a packet that has multicast address is sent to all interfaces which have become the members of the corresponding multicast group.

4.5 Media Access Control

The MAC address belongs to each node which is a part of a network. The data stacks are fed into a device node using the MAC address. Media refers to physical media, fibre or wireless that one uses by a device or node to get connected to the Internet. The nodes sharing the same physical network and IP address can make use of it. IoT device or sensor or actuator can be defined as node, the data-link layer of which is connected to the Internet. MAC address is 48 bit.

Each network card or Ethernet protocol involving a communicating node possesses unique MAC addresses of the source and destination node addresses. Ethernet frame is sent in the way of source-node MAC address of data stack and destination-nodeMAC address. The firmware in the network card or chip or core will contain the MAC address of each node.

The MAC address can address every node with usage of ARP. A router with an IP address transmits the network data to the node. RARP usage facilitates rendering every node to transmit on the network the data to a router with an IP address.

Address Resolution Protocol (ARP) involves a look up table. That table provides MAC address of the individual node using the network 32-bit address. That look-up is also

used in RARP. Authors save the IP address in the table in one column. MAC address is in

another column in each row. Rows will be as many as the number of nodes that are connected to the Internet. A single node MAC address gives the network a 32-bit address into the Internet with the use of the table. The lookup table is a table where the predetermined value contained in a column in a row of the table is used as the key to search a different value or a group of parameters in a different column(s) of the row.

ARP cache is a collection of the table concerning address translation of the data stack introduced to the Internet layer to the node MAC address. They store a cache known as ARP cache whenever it is the initial time that the network is transmitting the recipient IP and MAC addresses, or it is the initial time a node is transmitting the sender MAC and IP addresses. The cached addresses construct a look up table to the imminent communication amid the node and the system. The address resolution is achieved by allowing the node to send and receive the IP packets at the new IP address out of the Internet using the process.

Alternatively, the address can be stored in a local memory card or flash memory of a board or microcontroller chip. See this address when the microcontroller begins. The chip or the board connect to Ethernet LAN utilizes the the saved MAC address. The chip links the Internet with the help of this address and RARP. IP data stack is passed into the chip through ARP.

4.6 Summary

This chapter developed the principles of Internet-based connectivity systems, explained the communication of the IoT and connected devices based on Internet protocols, header, packet, segments, and network interfaces. It discusses both the structure and operation of the IPv4 and the IPv6 protocols, such as the creation of packets, addresses, routing, and how the TCP/IP layers work to encapsulate and process information as it goes through the communication stack. Low-power communication technologies of the IoT, including RPL of the lossy network and 6LoWPAN of the IPv6 communication on the IEEE 802.15.4-based devices with a focus on fragmentation, header compression, and constrained-node routing, are also discussed in the chapter. It also describes the IP addressing (static, dynamic, DNS, DHCP), the type of IPv6 address, the meaning of MAC address, and ARP/RARP mapping between IP and hardware addresses, and how this feature makes the IoT systems reliable in identifying and transferring data through the net.

Deep Science Publishing, 2025
https://doi.org/10.70593/978-93-7185-106-0

DeepScience
Open Access Books

Chapter 5: Modern Methods for Data Acquisition and Analytical Processing

5.1 Introduction

The new element of contemporary organizations is the data that has become the central asset of the organization and is used to make decisions, automate operations, apply artificial intelligence, and business intelligence. Nevertheless, data can only be useful after its adequate acquisition, organization, processing, and analysis. This chapter describes the entire cycle of data, data collection all the way to deriving valuable insights

5.2 Data Acquiring

Data acquisition may be described as systematic gathering of raw information of different internal and external sources. It establishes quality, completeness and the accuracy of all other subsequent stages..

5.2.1 Importance of Data Acquisition

- Ensures reliable inputs for decision-making
- Enhances model accuracy in analytics and AI
- Enables real-time monitoring and automation
- Supports organizational strategy and research

5.2.2 Types of Data Sources

1. Primary Sources

Data directly collected by the organization:

- Surveys, interviews
- Experiments
- IoT devices and sensors

- Manual observation

2. Secondary Sources

Data collected by external entities:

- Government data
- Research datasets
- Social media
- Web content

5.2.3 Methods of Data Acquisition

- Manual Data Entry – forms, spreadsheets
- Automated Collection – scripts, bots, system logs
- APIs & Web Services – cloud integrations
- Web Scraping – extracting data from websites
- Sensor-Based Acquisition – temperature, motion, GPS
- Streaming Data – real-time telemetry, IoT data

5.2.4 Challenges in Data Acquisition

- Incomplete or noisy data
- High volume and velocity (Big Data)
- Privacy and security concerns
- Sensor malfunction or transmission errors
- Ethical and legal restrictions (GDPR, HIPAA)

5.3 Data Organizing

After the data has been obtained, the process entails the correct organization of such data to ensure searching, managing and use.

Organisation of data is the ability of organising, structuring, categorising and storing data systematically to allow access, understanding, processing and analysis in an efficient and efficient way. Organizations of the new on-digital world deal with vast volumes of data that is generated in different sources. Unless it is well organized, the data turns out to be inefficient, lopses, and inaccurate. Data organization is a tool of qualified data management as well as analytics, machine learning, and decision-making.

5.3.1 Need for Organizing Data

- Improves data accessibility
- Reduces redundancy
- Ensures consistency and accuracy
- Enhances query performance
- Enables analytics and machine learning

5.3.2 Types of Data Based on Structure

Structured Data

- Highly organized
- Stored in rows and columns
- Example: SQL databases

Semi-Structured Data

- Flexible schema
- Includes tags or markers
- Example: JSON, XML

Unstructured Data

- No predefined structure
- Example: images, videos, audio, emails

5.3.3 Data Storage Models

1. File Systems – CSV, TXT
2. Relational Databases – MySQL, Oracle, PostgreSQL
3. NoSQL Databases – MongoDB, Cassandra
4. Data Warehouses – Snowflake, Redshift
5. Data Lakes – Hadoop HDFS, Azure Data Lake
6. Cloud Storage – AWS S3, GCP Storage

5.3.4 Data Cleaning and Preparation

Essential steps include:

- Removing duplicates
- Handling missing values
- Normalization and standardization

- Outlier detection
- Data transformation
- Integration from multiple sources

5.4 Data Processing

Data processing is a process that converts raw and unstructured information into useful and high quality data.

Data processing denotes the method of transforming raw and unstructured data to useful usable information which is organized and meaningful. It entails a process of activities such as gathering, authentication,

processing, data analysis, and data storage.

Nowadays, data processing is critical to business intelligence, machine intelligence, automation, and decision-support systems because of the data-driven world. Modern technologies have revolutionized data processing as previously it is done manually, and is now automated, real-time and cloud based pipelines capable of processing huge masses of data..

5.4.1 Stages of Data Processing

1 Collection – gathering raw data
2 Preparation – cleaning and filtering
3 Input – entering data into systems
4 Processing – applying algorithms
5 Output – charts, tables, reports
6 Storage – databases, warehouses

5.4.2 Types of Data Processing

1. Batch Processing

The term batch processing is used to describe a set of executing of jobs on some data set in an automated fashion. Information is stored throughout a period of time and done once using automated processes or scripts.

Batch Processing Workflow.

- Data Collection

Raw data can be collected and are in the form of logs, transactions, sensors or files.

- Data Staging

- The data is then stored in a temporary stage area where data is processed.
- Batch Job Creation

An analysis program is created as a batch script to work on the acquired information.

Example: payroll system

2. Real-Time Processing

A real time processing is the method of processing data in such a way that processing data is recorded, processed and a response is given in real time or within milliseconds. In contrast to batch processing, which uses great information and acts afterwards, real-time processing is an immediate information and response. It is important to note that it is urgent in applications that need a quick reaction, like fraud detection, online transactions, IoT monitoring, and live analytics.

Example: fraud detection

3. Distributed Processing

Distributed processing is a process that involves performing computational activities using multiple machines which are linked through a network. In place of using one powerful computer, a team of computers are used, to efficiently process high volumes of data.

Distributed Processing Architecture

1. Distributed Nodes

A group of computers performing coordinated tasks.

2. Master/Coordinator Node

Assigns tasks, manages workers, tracks progress.

3. Worker Nodes

Execute tasks like:

- Data processing
- Storage
- Aggregation

4. Distributed Storage

Data is split into chunks and stored across nodes (e.g., HDFS, Google File System).

5. Network Communication

Enables nodes to exchange data and results.

Tools: Hadoop, Spark

4. Cloud Processing

Cloud processing is a method of handling data and computational workloads using distributed, virtualized cloud infrastructure. Processing tasks—such as data cleaning, transformation, analytics, and AI operations—are performed on cloud-based servers that scale dynamically based on demand.

Architecture of Cloud Processing

1. Cloud Infrastructure

Includes compute resources like:

- Virtual Machines (EC2, Compute Engine)
- Containers (Docker, Kubernetes)
- Serverless compute (AWS Lambda, Azure Functions)

2. Storage

Cloud platforms provide scalable and durable storage:

- Object storage (S3, Azure Blob)
- Distributed file systems
- Cloud databases (DynamoDB, BigQuery, CosmosDB)

3. Processing Engines

Cloud environments support:

- Big data processing (Dataproc, EMR, HDInsight)
- Stream processing (Kinesis, Dataflow)
- Machine learning platforms (SageMaker, Vertex AI)

4. Network & Security

- VPC/VNet for private networking
- Identity and access management (IAM)
- Load balancers and firewalls

Example: AWS Lambda, Google BigQuery

5.4.3 Techniques in Data Processing

- Sorting & filtering

- Aggregation
- Encoding & decoding
- Data transformation
- Feature extraction
- Statistical computations

5.4.3.1 Sorting & filtering

Sorting and filtering are two basic data processing commands that enable an analyst to reorganize datasets, discover patterns and draw intelligent conclusions effectively. The operations have customer applications in databases, spreadsheets, data analysis software, and programming environments. Sorting tables the data in a logical arrangement whereas filtering isolates a data set according to given requirements.

Sorting is the process of organizing data elements in a pre-determined order, i.e. ascending, descending, alphabetical, numeric or chronological..

Objectives of Sorting

- To enable quick searching
- To group related data
- To prepare data for advanced computations (e.g., binary search, aggregations)
- To improve readability and organization

5.4.3.2 Aggregation

Another important data processing operation that can be performed on data is aggregation which is the action of the combination of several data values into one meaningful summary. It assists in transforming the information on records captured in detail to a higher level of insights that can be used in analysis, reporting and decision making. Aggregation has common applications in databases, spreadsheets, business intelligence (BI) tools and programming.

5.4.3.3 Encoding & decoding

Encoding and decoding are some of the core processes in the communication systems and data processing. When a piece of data is converted into a different form, this is called encoding, typically to be stored, transmitted, or interpreted efficiently; conversely, conversion of data into its original form is called decoding, and thus reconstructing and recovering the original data. They are used together to ensure that data is represented accurately, securely or in a machine readable form.

5.4.3.4 Data transformation

Data transformation is a transformation of data according to its form or structure into one more suitable to be analyzed, stored, or processed. It is an important procedure of preprocessing data and is a popular operation in ETL (Extract-Transform-Load) processes, in data warehousing, machine learning pipelines, and data analytics.

5.4.3.4 Feature extraction

The concept of feature extraction involves the extraction of raw data to a list of meaningful, informative and compact features which can be effectively applied during analysis, classification, clustering or prediction. It converts multidimensional data into one-dimensional form and the attributes that are important to it are retained, therefore enhancing efficiency in computation and the performance of the model.

Machine learning, pattern recognition, image processing, speech analysis and natural language processing are among the other fields of use of feature extraction.

5.4.3.5 Statistical computations

Statistical calculations form a very significant step in the data processing procedure as it allows the analysts to summarize, interpret, and extract valuable facts out of the raw data. The computations are useful in determining patterns and trends and relationships that aid in making business, science, engineering, and machine learning decisions.

The broad categories of statistic computation are measures of central tendency, dispersion, correlation, probability distributions and hypothesis testing. This is aimed at transforming number data to meaningful measures of the data that define the nature of the data and its variation.

5.5 Data Analytics

Data analytics is concerned with deriving meaningful insights, trends and making decisions..

5.5.1 Goals of Data Analytics

- Improve operational efficiency
- Predict future outcomes
- Discover hidden patterns
- Support real-time decision-making
- Reduce risks and enhance strategy

5.5.2 Types of Data Analytics

1. Descriptive Analytics

Explains what has happened

Tools: , summary reports

2. Diagnostic Analytics

Explains why it happened

Tools: drill-down, root-cause analysis

3. Predictive Analytics

Forecasts what will happen

Tools: ML models, regression

4. Prescriptive Analytics

Recommends what should be done

Tools: optimization, simulations

5.5.3 Tools and Technologies

- Python (NumPy, Pandas, SciPy)
- R Programming
- Tableau, Power BI
- Hadoop, Spark
- SQL Analytics

5.5.4 Visualization in Analytics

Definition

Data visualization refers to a process where data is displayed in charts, graphs, plots, dashboards or other interactive monetary forms to aid in analysis, decision-making and communication.

Importance of Visualization in Analytics

- Enhances comprehension of large datasets
- Identifies patterns, trends, and anomalies quickly
- Facilitates comparison across variables
- Supports informed decision-making
- Communicates insights effectively to non-technical audiences
- Enables real-time monitoring of processes and KPIs

Common charts:

- Line chart
- Bar chart
- Pie chart
- Heatmap
- Scatter plot
- Dashboards

5.6 Applications

Healthcare – disease prediction, medical imaging

Finance – fraud detection, credit scoring

Manufacturing – predictive maintenance

Retail – customer segmentation

Agriculture – crop monitoring

Smart Cities – traffic and pollution analytics

5.7 Challenges and Ethical Issues

- Data quality issues
- Privacy and confidentiality
- Bias in analytics and AI models
- Data governance constraints
- Security threats

5.8 Summary

This chapter was a summary of all data lifecycle and includes the process of data acquisition, data organization, data processing, and data analytics. Organizations will be able to extract raw facts and create meaningful information and forceful choices by a firm base throughout these levels. These ideals are the foundation of data science in modern times, IoT, business intelligence, and innovation that is driven by AI.

Deep Science Publishing, 2025
https://doi.org/10.70593/978-93-7185-106-0

DeepScience
Open Access Books

Chapter 6: Smart Data Collection and Cloud-Based Computing

6.1 Introduction

IoT Information gathering refers to the act of collecting information of Internet of Things (IoT) systems, sensors and devices. These gadgets continuously or periodically record data on the environmental conditions, device conditions, motion, position or user action. The data obtained is then sent to a gateway, cloud platform or a server where data are stored, processed and processed. Smart homes, healthcare, industry and transportation IoT data collection can be used in real-time-monitoring and automation as well as decision-making applications.

Some of the traditional ways of collecting data and data storage are as follows:

- Registering device generated data on a local server which serves the devices in the locality.
- Moving/storing data of the devices used in the form of local files that are put in removable storage devices like micro-SD cards or hard disk on a computer..
- Sending and storing data, along with computation results, in a locally maintained data store or coordinating node.
- Saving device data at a local node that operates as part of a distributed database management system (DBMS).
- Forwarding and storing data at a remote node within the distributed DBMS architecture.
- Transmitting data over the Internet and storing it in a database on a web server or enterprise server.
- Sending data through the Internet to be stored in an enterprise-level data center.

6.2 Cloud-Based Framework for Data Collection, Storage, and Processing

Different methods of data collection, storage and computing

I. Devices or sensor networks data collection at the device web server,

II. Local files,

III. Dedicated data store at coordinating node,

IV. Local node in a distributed DBMS,

V. Internet-connected data centre,

VI. Internet-connected server,

VII. Internet-connected distributed DBMS nodes, and

VIII. Cloud infrastructure and services.

Cloud computing is one of the significant innovations in Information and Communications Technology (ICT). This paradigm uses data collection, storage, and calculations with help of XaaS (Everything-as-a-Service) products on cloud platforms that have Internet connectivity.

Learning the cloud computing platform requires that one familiarizes himself with the important terminologies and their meaning before proceeding further.

A resource can be defined as any computational object that assist in read, write (creation or modification) or execute processes in a system. Path specifications too are considered to be resources because they are addressable resources available in the execution of a program. Resource captures indivisible units of information which either can be consumed or manipulated in the process of computation. A resource could being in the form of a single instance or instantiations depending on the architecture of the system. Data points, memory pointers, structured data objects, persistent data stores and executable methods are all resources as they are addressable and operable elements of the computational environment.

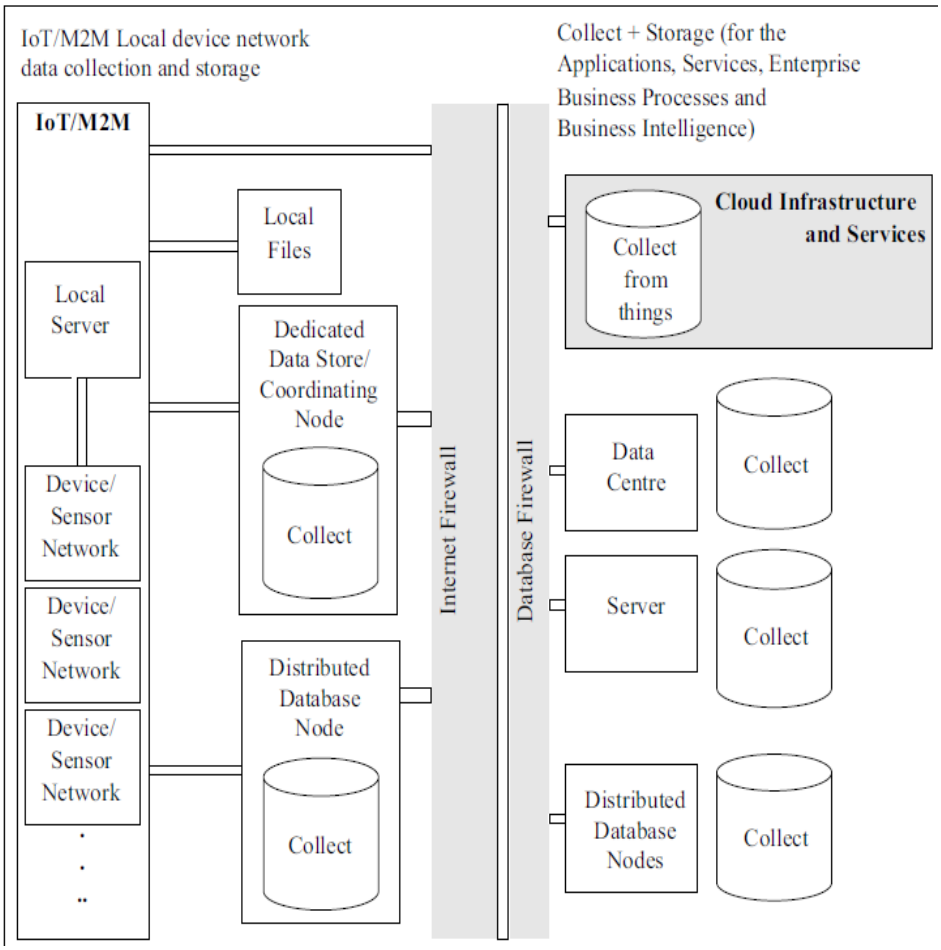


Figure 7Devices or sensors network data collection at a device local-server, local files, dedicated data

System Resource

- Since systems demonstrate a context specific to each system, a system resource can be any underlying computational component, the operating system, memory subsystem, network interface, server infrastructure, software component or application component that abstractly supports the execution of processes and system functionality.
- System resources are now the fundamental architectural components of a computing environment which would be OS services, memory units, networking components, server instances, and executable software artifacts.

Environment

- An environment is a configured environment, which aids program development, or program execution or both. It offers the required toolchains, interfaces and runtime mechanisms of application creation and application deployment. These examples are the online IDE Cloud 9 used in designing IoT applications using the BeagleBone, the windows execution engine used to program an application, and the Google App Engine used in the development of applications using either the Python or Java language.
- Environments are formalized computational environments comprising of development tools, runtime support and system services in support of software engineering processes and application life cycles.

Platform

- A platform refers to the combined base of software applications or services, hardware, and networking infrastructure upon which it is completed, built, and distributed.
- Platforms are the building block of computation by unifying physical hardware, OS-level services as well as communication layers around which higher-level software systems and services are constructed, deployed and executed.

Edge Computing

- Edge computing is a distributed model of computation that is installed on IoT nodes nearer to the place where the data is generated such that computation, data processing, and the delivery of the service should be provided by the periphery of the network not by the centralized cloud services..
- It shifts computational workloads to the logical edges of the network, enabling faster processing and reduced latency by operating near IoT data sources.

Distributed Computing

- *Distributed computing* denotes the execution and coordination of computational tasks across multiple geographically or logically dispersed computing resources interconnected through the Internet.
- It involves using a network of distributed environments to share, allocate, and process workloads collectively.

Service

- A service is a software model, a way of wrapping particular capabilities and aggregated functionality, which may be called by applications programmatically.

- Services are characterized by descriptive metadata and mechanism of discovery that include advertisements or broker based registries. They exist within Service Level Agreements (SLAs) that bind the endpoint of service consumer to the service endpoint.
- The service can also use or call other services in its functioning.
- W3C defines a web service as an application that is accessible through a URI, the description of which is done with the help of XML-based Web Services Description Language (WSDL) and can be identified with the help of the common-registrar systems.
- Web services can communicate with other applications, services via the exchange of the structured XML messages via Internet protocols that provide interoperability between heterogeneous systems.

Service-Oriented Architecture (SOA).

- Service oriented architecture is a design model whereby the system parts are designed as autonomously loosely coupled services, which can be dynamically bound, coordinated and re-prosille.
- Trade between these services is based on communication of messages. Orchestration refers to the directed order in another term, parallel or sequential, according to which the services are called and the flow of data between them.

Web Computing

- Web computing can be defined as the computational tasks that are performed out of the server resources or using the web services provided on the Internet by the web servers.
- It is based on the web technologies and server set ups to facilitate the execution of applications and processing of data.

Grid Computing

- In grid computing, the computations are made by utilizing a federated network of interconnecting and shared computing resources that constitute a grid, as opposed to using individual traditional web servers.
- It facilitates sharing of resources on large scale and high performance computing on dispersed infrastructures.

Utility Computing

- Utility computing is a service-oriented computing model whereby resources are n-demand and delivered according to the levels of services that are needed as well as supported by shared pool of computational environments.

- Using them is just like using utility services like power or water whereby, resources and services are consumed upon demand.

Cloud Computing

Cloud computing refers to a model whereby computational services are provided as a set of web-based services that are offered in the platform of a cloud service provider. This paradigm facilitates the use of the distributed, grid, and utility-style computation in the connected systems.

Key Performance Indicator (KPI).

A Key Performance Indicator (KPI) is a measurable value in the form of a single or several values that are being monitored like the minimum, average, and maximum values of performance and operational behaviour.

Localisation

Localisation means evaluating the use of cloud services based on Quality of Service (QoS) parameters and KPIs of the particular geographic or logical location the cloud resources are used.

Seamless Cloud Computing

Seamless cloud computing refers to continuous availability of cloud services that the execution of applications and the consumption of resources continues without causing disruption upon a change in location that supports similar QoS conditions and KPIs such as the developer using the same cloud platform after a relocation.

Elasticity

Elasticity defines the use of the ability of an application to provide and remove local and remote services or resources not only when demand happens but also dynamically. Costing of the users is based on the use of resources and the KPIs.

Measurability

Measurability of resource or service also implies that one is able to quantify a resource or service towards monitoring, control and reporting of the delivery or performance of that resource or service..

Homogeneity

Homogeneity in a cluster or multi-cluster computing environment refers to the presence of uniformly configured nodes with compatible kernels, enabling automatic migration of processes across nodes. System software on each node must ensure identical data representation and consistent computational outcomes.

Resilient Computing

Resilient computing denotes the capacity of a system to sustain the agreed-upon QoS levels and KPIs despite encountering various operational challenges, guided by well-defined resilience metrics. Such challenges may vary from minor misconfigurations to large-scale disruptions such as natural disasters. The concept aligns with the general notion of resilience—the ability to regain an original functional state after disturbance.

Scalability

Scalability in cloud services refers to the ability of an application to utilize both small-scale local resources and large-scale remote servers, expanding or contracting resource usage as needed. This adjustment occurs with cost proportional to the usage level as demand increases.

Maintainability

- Maintainability in cloud services refers to the management and upkeep of storage systems, applications, computing infrastructure, services, data centers, and servers, all of which are handled by the cloud provider. These maintenance responsibilities incur no direct cost to the end user.
- Cloud maintainability denotes the ability of the cloud provider to perform continuous updates, repairs, and optimizations of underlying infrastructure and services without requiring user intervention or expense.

XAAS (Everything-as-a-Service)

- *XAAS* represents an architectural paradigm that supports the development and deployment of applications by delivering functionalities as services through web technologies and service-oriented architecture (SOA).
- This computing approach integrates complex applications and distributed services, utilizing the XAAS model as the foundation for implementing cloud platforms.
- XAAS abstracts heterogeneous resources and application capabilities into service-based components that can be consumed on demand.

Multitenant Cloud Model

- A *multitenant cloud model* refers to a cloud environment in which multiple users or organizations share access to a common computing platform while being governed by individual Service Level Agreements (SLAs) specifying QoS and KPI requirements.

- In this model, resources are pooled and shared among tenants, but each user is billed independently according to the agreed-upon performance and usage metrics.
- Multitenancy ensures efficient resource utilization while maintaining logical isolation and customized service levels for each tenant.

6.2.1 Foundations of Cloud-Based Computing Models

There is cloud computing, which involves a set of services found on the Internet. The computational functionality is provided by cloud. Cloud computing makes use of the infrastructure of cloud-service provider. The infrastructure implements on a utility or grid computing or webservices environment, which consists of network, system, grid of computers or servers or data centres.

Cloud Service Platform Framework

Cloud platform offers the following:

I. Large data storage infrastructure of large devices, RFIDs, machine industries plant, cars and networks of devices.

II. Computational capabilities, e.g. analytics, IDE (Integrated Development Environment).

III. Teamwork computing and sharing data store.

Cloud platforms are utilized to enable integrated connectivity among heterogeneous devices, distributed data sources, application programming interfaces (APIs), software applications, and service components. They support interaction between users, enterprises, and business systems while providing an operational foundation for delivering Everything-as-a-Service (XaaS). These platforms function as unified environments that coordinate device communication, data exchange, service orchestration, and application deployment within large-scale distributed computing ecosystems.

Internet Cloud + Clients = User applications and services with 'no boundaries and no walls'

A platform comprises of hardware, network and operating system (OS) upon which an application or service is executed. Further, various applications can be initially developed to be deployed on diversified platforms (OSs, hardware and networks). They should be unified on a single platform and base of applications and services.

Cloud storage and computing environment avails virtualised environment which is defined as an existing environment, and it is presented to look like a single environment, but in reality two or more running environments and platforms may exist physically..

Cloud Computing Features

- On-demand self-service for provisioning compute, storage, and software resources
- Resource pooling in a multi-tenant architecture
- Broad network access for heterogeneous clients and devices
- Elastic resource scaling
- Large-scale availability
- High scalability for varying workloads
- Centralized maintainability handled by the provider
- Homogeneous computing environment for consistent performance
- Virtualization for flexible and isolated resource allocation
- Interconnected virtualized platform supporting enterprise SLAs
- Resilient computing with fault tolerance
- Enhanced security mechanisms
- Cost efficiency through pay-as-you-use model

Cloud Computing Concerns

- Requires stable, high-speed Internet connectivity
- Service feature limitations depending on provider
- Risk of data loss or corruption
- Possible failure to meet SLA-defined performance
- API and protocol differences across cloud providers
- Security risks in multi-tenant environments
- Reduced user control over data and infrastructure

6.2.2 Types of Cloud Deployment Models

Public Cloud

- A public cloud is a deployment model in which cloud infrastructure is provided by commercial vendors, educational institutions, government bodies, or enterprises and made accessible to the general public.
- Public clouds offer computing resources to anyone who wishes to use them, typically on a subscription or pay-per-use basis.

Private Cloud

- A private cloud belongs to a single entity - enterprise, institution or industry and is available to its employees and authorized persons.
- This model has improved control, security and customization as the infrastructure is dedicated to the internal organizational use.

Community Cloud

- A community cloud is one that is provided to a particular community of organizations sharing similar needs, i.e. security policies or compliance and missions.
- It is used by collaborative community created by various institutions or enterprises and can only be accessed by the community participants as well as users related to them.

Hybrid Cloud

- The hybrid cloud involves the combination of two or more distinct cloud structures such as public, private or community; each having their own data storing and application; yet integrated to one another through proprietary or standardized technologies..
- This model enables workload portability, data exchange, and resource integration across multiple cloud environments.

Examples of cloud platforms are Amazon EC2, Microsoft Azure, Google App Engine, Xively, Nimbits, AWS IoT, CISCO IoT, IOx and Fog, IBM IoT Foundation, TCS Connected Universe Platform.

6.3 Cloud Connectivity and XaaS Concept

- Cloud computing interconnects devices, data sources, applications, services, users, and business systems within a unified digital ecosystem.
- Cloud services function as *distribution services*, enabling coordinated access to computing resources such as processing capabilities, storage systems, networks, servers, and applications.

The cloud computing paradigm can be conceptually represented as:

Cloud Computing = SaaS + PaaS + IaaS + DaaS,

Indicating that cloud services integrate software, platform, infrastructure, and data delivery models.

Software as a Service (SaaS)

- SaaS provides on-demand access to software applications that are hosted and managed within the cloud.

In the SaaS model, applications run on cloud infrastructure and are delivered to users over the Internet as needed.

All aspects of software management—including updates, upgrades, maintenance, security, and infrastructure support—are handled entirely by the cloud service provider.

SaaS enables users to access applications without installing or managing local software environments.

Platform as a Service (PaaS)

- PaaS offers developers an on-demand cloud-based platform for building, testing, deploying, and managing applications.
- The PaaS environment supplies computing resources, storage, development tools, database services, and application-hosting platforms accessible over the Internet.
- The cloud provider is responsible for maintaining the platform, updating system resources, ensuring security, and supporting developer-specific requirements.
- With PaaS, developers can focus on application logic while infrastructure and platform management are abstracted away.

Infrastructure as a Service (IaaS)

- IaaS provides virtualized computing infrastructure—including servers, networks, storage systems, and data centers—on an on-demand, pay-as-you-use basis.
- Developers or users install their preferred OS images, databases, and applications on the rented infrastructure and maintain full control over the deployed software environment.
- The cloud provider manages the underlying hardware, virtualization layer, networking, and physical security.
- IaaS supports scalable infrastructure provisioning within a multi-tenant environment where resources are consumed as needed.

Data as a Service (DaaS)

- DaaS delivers data stored in cloud-based data centers to applications or users on-demand via the Internet.
- Under the DaaS model, enterprises access data repositories or data warehouses through pay-per-use mechanisms.

- The data center provider ensures continuous availability (24×7), manages power, replication, mirroring, scaling, security, and infrastructure maintenance.

DaaS enables organizations to use centralized cloud-managed data without needing to maintain their own physical storage infrastructure.

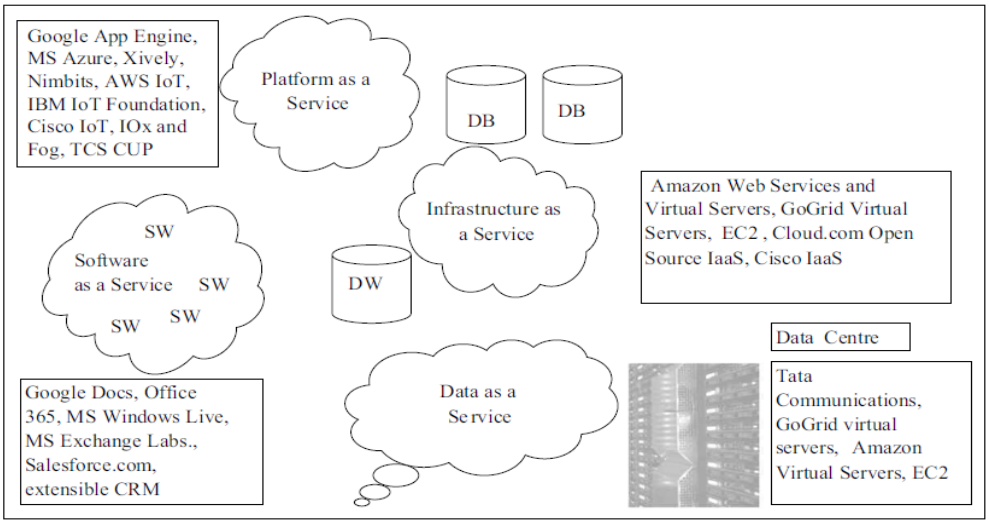


Figure 8 PaaS, SaaS, IaaS and DaaS Cloud Service model

6.4 Cloud-Enabled IoT Services Using Xively, Nimbits, and Related Frameworks

An application or a service is a user. The feeds or responses are received by the user of the application or service. A service based on IoT cloud offers a data collection, data points, messages and calculation objects. The service also offers to generated and communicated alerts, triggers and feeds to the user. The deployment of a server at the edges (device nodes) and relaying the feeds to the cloud service is possible.

A data point is produced each time a sensor records a new measurement or sensed value. A *feed* refers to a collection of data points, objects, data streams, or messages that are generated and transmitted after applying filtering, computation, compaction, fusion, compression, analysis, or aggregation rules. Feeds may also consist of alerts triggered by predefined thresholds, programmed events, or alarms configured within the system. A feed can represent a continuous data stream generated and transmitted at current, real-time intervals. Feeds are generated for consumption by an application or service that requires specific information for processing or decision-making. Feeds provide the necessary data instances at predefined intervals, delivered to an application or service through push, publish–subscribe, or other communication modes. Feeds are structured outputs in IoT systems, which are used to represent processed information in ways that

are efficient to pass data to applications or cloud-based services. Feeds can contain both unprocessed and processed information based on the needs of the system, so the valuable information can reach the subscribed users or services in time.

6.4.1 Data collection, storage, computing using Xively based on IoT Cloud.

Pachube was a Web-based application that was meant to be used in real-time data acquisition and sharing via the Internet. This platform was renamed as Cosm, and the introduction of the console-based interface in the management and monitoring of the data feeds occurred. Its further development resulted in the domain name of Xively, which is the most recent release of the service. Xively was also an open-source system that was compatible with Arduino, an open-source prototyping platform that allows the devices to connect to the Internet. Xively was a commercial IoT/M2M Platform-as-a-Service (PaaS), which provided support of data aggregation, data mining, and real-time data exchange of Web of Things. Xively offer cloud PaaS that was focused on IoT based services and business applications. This platform had a support provision of various communication protocols like REST, WebSocket and MQTT that would have an easy access to Xively Cloud Services.

It was also referred to as native Android, Arduino, ARM mbed, Java, PHP, Ruby and Python cross-platform software development kits. The toolchain offered by Xively with the built-in provision of the full lifecycle of an IoT project (prototyped to deployed and the further maintenance of the devices) could be actively used by the developers. The ecosystem was developed in such a way that it made it easy to design a connected product with standardized APIs, device libraries and cloud integration services.

Xively PaaS service features are the following:

I. Xively offers a business and service platform that connects Internet with hundreds of products, i.e. collaboration tools like Rescue, BoldChat and join.me, etc.

- It helps gather data in real-time of the connected equipment on the Internet.
- The platform will provide data visualization to sensor output and measurements of the IoT devices.
- Graphical plots can be created by the users to evaluate and analyse the data that has been gathered more efficiently.
- The system can emit alerts depending on rules which are specified or events that are detected.
- It gives access to past data, which allows one to analyze trends and track long-term trends.
- The Xively product has Java, Python, Ruby and Android development environments.

- The platform has the capability of formulating feeds representing the real-life or virtual objects as defined by system and user-defined object.
- It includes IoT devices based on ARM mbed, Arduino and a wide range of other hardware platforms plus offers straightforward HTTP-based APIs, so it is simple to have device hardware serve as a client and to communicate with Xively web services and send data..
- Xively is fully compatible with RESTful communication interfaces.

A user must create an account on Xively before deploying its APIs for data collection and related functionalities. During configuration, the API key available under the user's account settings must be copied and used for authentication. Xively APIs support interfacing with technologies such as Python, HTML5, HTML5 servers, Tornado, WebSocket, WebSocket servers and RPC (Remote Procedure Call) mechanisms. By using these APIs, devices gain a web-accessible presence, enabling remote interaction and monitoring.

For instance, an Arduino-based climate monitoring client can be made accessible through a web browser or mobile application using Xively services.

Arduino is an open-source prototyping platform built around ATmega microcontrollers, widely used for embedded systems and IoT/M2M applications. Another supported platform is mBed™, which is based on ARM Cortex microprocessors and similarly designed for embedded and IoT/M2M solutions.

Xively operates as an open-source platform that allows IoT devices and sensor networks to stream sensor data to the web. The platform offers capabilities for logging, sharing and visualizing diverse sensor datasets using an HTTP-based API. With its APIs, Xively makes it easy to integrate devices and provides easy communication between the devices nodes and cloud-based platforms.

Xively works around such core entities as users, feeds, data streams, data points and triggers.

A feed usually corresponds to definite physical place, e.g. home, whereas data streams are different sensors which can be found at this place i.e. temperature devices, detecting energy consumption or light in the room.

Pull or Push Techniques of IoT Devices.

Xively has two main collecting device data mechanisms: a pulling mechanism, or one whereby data is automatically pulled into an HTTP server, and a pushing mechanism where an HTTP client specifies the transfer of data to the platform manually.The pull mode functions as an automatic feed type, whereas the push mode enables users or devices to directly write data to Xively.

Data Formats and Structures

- Multiple data formats and data structures are supported to facilitate communication, data collection and service integration with Xively.
- The platform provides compatibility with JSON, XML and CSV formats, enabling structured and interoperable data exchange.

Private and Public Data Access

The free Xively account supports up to ten sensor feeds which can be updated near in real time and retains the data within the account up to three months.

- The system provides interactive graphical user interfaces which can be incorporated into mobile app enabling users to show the sensor data anywhere.
- The data feeds generated by other users may also be incorporated into applications making shared or open datasets to be the input to new services.
- Xively has built both private and public access modes so that a developer can control access and sharing of sensor data.
- Xively promotes the usage of the data streams, the data points and triggers as the fundamental elements of its data model. A data stream means the flow of sensed data consisting of the perpetual flow of information conveyed by a sensor or a device using the Internet.

Data points are the individual values of measurement at given times of time. A trigger is an automated activity that is triggered whenever a specific condition or change in state occurs. An example is that a trigger can be sinisterly evoked with a decrease in ambient light, which goes beneath a predetermined limit around a group of streetlights, or a data item surpasses or decreases a defined maximum, minimum or target. Allowing the rule-based automation triggers are used to observe incoming data points and perform prescribed responses to these data points at the meeting of threshold conditions. The sensor readings are grouped into streams of data, arranged in their time sequence which allows real-time analysis of sensor changing behaviour to be easier.

Creating and Managing Feeds

- A cluster of streetlights could be sensor network or groups of IoT devices that can send environmental information to Xively using the Internet. All individual devices can support data streams of their own, in terms of sensed parameters, to Xively.
- It is possible to create and control feeds to organize incoming data of various sensors or a system of IoT nodes. Xively gives users the ability to format and maintain feeds that combine information streams of different interconnected

items. Devices are connected to Xively through the Internet (Stack) where they send its measured values and feed them in the centralized feed management.

Visualising Data

- Xively allows real-time data capture using the Internet as well as graphing capabilities, creation and retrieving of historical records. The platform also has advanced data visualization features on all the feeds as well as individual streams of data.
- Xively allows users to track, graph and calculate sensor data in interactive visual formats. Depending on the device setting, the system supports the manually created feeds and automatically created feeds. Real-time visualization tools as offered by Xively are useful to users who wish to interpret sensor behaviour, trend identification and responding to the alarm effectively.

6.4.2 Nimbits, is an IoT cloud computing system which offers data collection and real-time processing capabilities of distributed sensor networks.

Nimbits allows IoT applications by means of a free-source, distributed cloud platform.

Under the Nimbits cloud PaaS model, the instances of the Nimbits Server are deployed at device nodes so that the local processing can occur.

As a data store to be used as an M2M, Nimbits is used to collect and log data and retrieval of historical sensor values.

It has an architecture based on cloud technologies, such as the integration based on Google App Engine to allow scale deployment.

Nimbits server is deployed as a class hierarchy in Java hierarchy, that is, com.nimbits.server.system. ServerInfo in the Object engineering of Java.

Nimbits PaaS Features

Edge and Distributed Computing

- Nimbits supports edge computing by running local applications on embedded systems, executing rules close to the source of data.
- When Internet connectivity is available, important or filtered data is pushed upward to the cloud-hosted Nimbits Server instance.
- Device-level instances of Nimbits Server facilitate decentralized processing before cloud synchronization.

Programming Language and Integration Support

- The platform supports a wide range of programming languages, including Arduino (with updated libraries), JavaScript, HTML, and the Nimbits.io Java library.
- Arduino devices can push data directly to the Nimbits cloud through native libraries and API functions.

Backend and Data Handling

- The Nimbits server acts as a backend data platform, enabling data points to relay values between software systems and hardware devices such as Arduino.
- The open-source nimbits.io Java library simplifies development of Java, Web, and Android applications that consume Nimbits data, notifications and messages.

Rule Engine and Automation

- Nimbits includes a built-in rule engine that links sensors, applications and users to the cloud and to one another.
- Rules have the capability to perform calculations, statistical operations, email notifications, XMPP messages or push notifications.

Data Logging and Storage

- The platform also offers long-term data logging services which store the data points and data objects which can be accessed with time.
- Nimbits data points can hold any type of data, which is serializable, e.g., a json or XML.
- Noise filtering reduces the amount of data transmitted to central or high level Nimbits instances to important data changes only.
- The platform has the ability to break down specialized data and store them effectively.
- Data may be time or geo-stamped in order to make contextual use of them..

Visualization and Real-time Access

- Nimbits clients support real-time data collection, charting, graphing and manual data entry over the Internet.
- Visualization tools allow users to monitor sensor measurements and IoT device data through charts and graphical interfaces.
- Alerts and Event Handling
- Nimbits supports real-time alert subscription, generation and dissemination over the Internet.

- Streams of data objects can be created and stored as part of a data point series.

Remote Accessibility and Device Control

Users can access and monitor data from any location, enabling remote adjustments to the behaviour of connected devices and software.

Hardware Platform Support

- The platform supports major IoT hardware ecosystems, including mBed™, Arduino and Raspberry Pi-based devices.
- Lightweight web service APIs enable device hardware to communicate with Nimbits cloud services and transmit data seamlessly.

Deployment Options

Nimbits instances can be deployed on Google App Engine, J2EE servers running on Amazon EC2 or even compact systems such as Raspberry Pi.

Data Points

A data point represents the value captured by an individual sensor within a larger group of sensors. Data points help structure and organize sensor information in various hierarchical or logical forms. In Nimbits, a data point can contain subordinate data points, often referred to as child points. Child points function as sub-elements of a primary data point; for example, if “light level” is the main data point, then “light ON/OFF” could serve as a child point.

Additional child points may represent threshold-based conditions, such as whether the light level is above or below a specified limit. Such a hierarchical grouping of data points allows much richer context and finer granularity over tracking sensor behaviour. Data point can then be used to represent not just the raw values but also derived states or conditions as the other points that consist of the child points. The grouping of sensor data into parent and child data points enhances the clarity of sensor data, event processing and rule-based automation of IoT systems.

Data Channels

A user has the opportunity to create data feed channel that shows the events and messages and subscribed alerts in the system. The channels enable the user to observe real time alert notifications created by subscribed data points. Users can also subscribe to any data point added by other users, and subscribe it in a way that any subsequent information will be added in their feed.. Through the data channel, users can observe states such as idle, high or low alerts as they occur. A user’s data feed operates as a Nimbits data point, providing a centralized location for receiving and viewing event

updates. Data channels function as communication streams where sensor alerts and system messages are aggregated for easy monitoring. Subscriptions can be customized to relay specific types of alerts or data conditions directly to the user's channel. Because data feeds are implemented as Nimbits data points, they integrate seamlessly with the platform's overall data model.

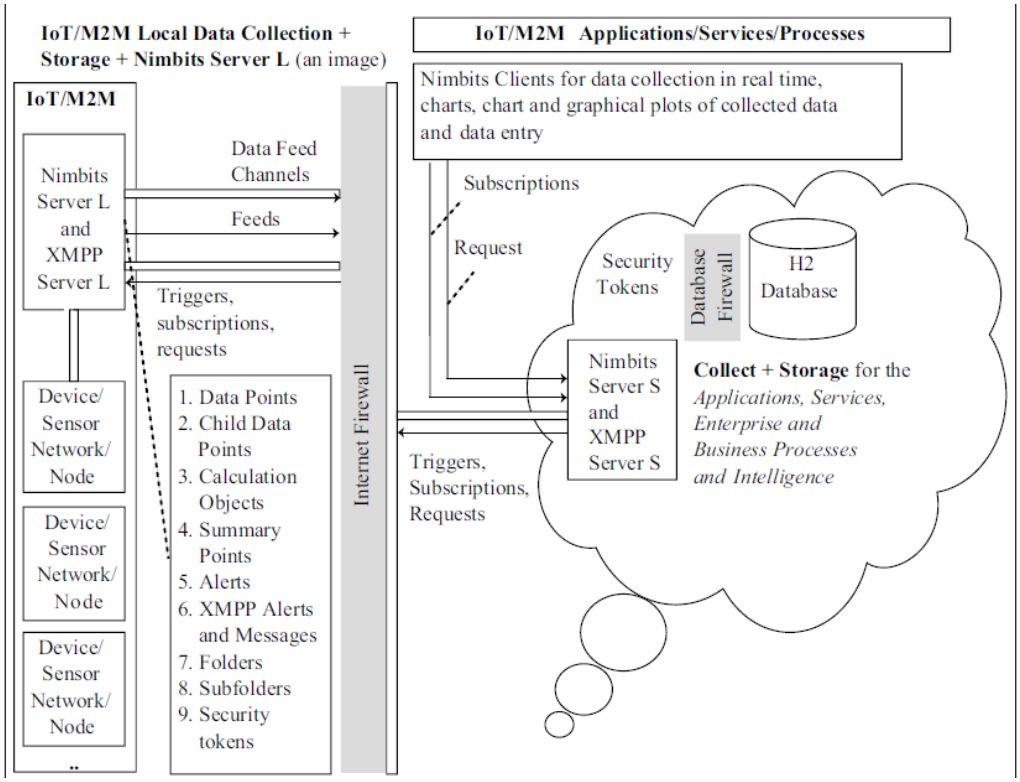


Figure 9Connected devices, sensor nodes, network data points, Nimbits server, deployment at the device network nodes, and networked with the Nimbits Server (PaaS, SaaS and IaaS services) at cloud for applications and services.

6.5 Summary

This explains how IoT devices collect, store and process data using cloud- based frameworks. It introduces core computing concepts such as resources, platforms, services, edge computing and cloud models. The chapter describes cloud features and deployment models, including SaaS, PaaS, IaaS and DaaS. It outlines how data points, streams, feeds and triggers enable real- time IoT monitoring and automation. Xively and Nimbits are presented as examples of cloud IoT platforms that support data acquisition, visualization, storage and rule-based processing.