IoT Application Development (4 of 14)

IBM

IBM ICE (Innovation Centre for Education)

Infrastructures of IOT devices

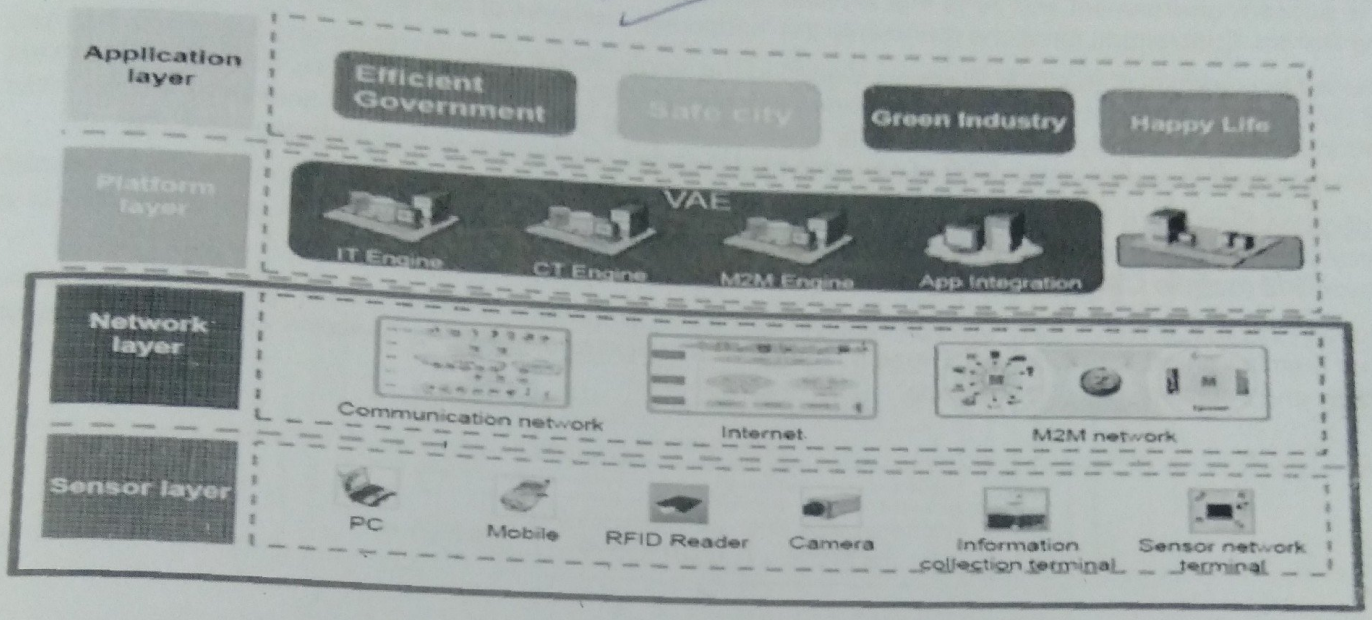


Figure 3-16. IoT Application Development (4 of 14)

Notes:

IOT Platforms: Traditionally, The Internet of Things, is rapidly evolving state of computing where everything are connected from refrigerators to energy grids to the Internet and communicating with each other, will create new challenges for data centers. Today, IOT solutions are doing many functions than just monitoring the status of remote assets and equipment. They are gathering the real-time data from millions of connected machines, for example, tractors, medical devices, vending machines, storage tanks, and translating it into meaningful information for the quick decisions, automated actions, and strategic analytics. Solutions such as software-defined networking (SDN) promises to help IT teams to manage converged network infrastructure easily and centrally, and also allows for dynamic, massive scalability and elasticity of data centres to support Internet of Things and its applications, and enabling automation of such tasks as configuration and policy management.

The primary driver for IOT solutions is now enabling the new services, rather than just improving the operational efficiency/cost saving. A platform is considered to be group of technologies that are used based upon which applications, processes or other technologies are to be developed/delivered. Creating a platform is usually the complex and delicate task, that needs to serve multiple purposes, but primary purpose is to support and simplify the work of those who will be using/consuming this platform. IOT platforms have transformed the IOT market by making device data more accessible to the application developers, and also by offering the well-defined software interfaces and making APIs are available so that application developers can readily integrate the information sources and also control parameters into their applications.

Types of IOT Platform: Over the past decade, the IOT platform space has developed rapidly, and now that includes the following broad platform functions are:

Connectivity Support: It encompasses all the most fundamental tasks that are to be undertaken to configure and supports the machine-to-machine connection. In a mobile environment, the tasks that include connection provisioning, usage monitoring, and the some level of support for the fault resolution.

Service Enablement: It has the extensive capabilities in terms of the solution support, reporting and provision of a software environment and APIs that facilitate solution development. Together, the Connectivity Support and Service Enablement functions represents the 'horizontal' elements of the IOT platforms industry.

Device Management: It has typically been aligned to the single device manufacturers and potentially supports the devices of multiple types and vendors are connected through the multiple networks. Device management platforms that essentially exists to facilitate sales of devices (and device-centric solutions) where these devices typically requires some form of non-standard systems support (reporting, management, etc).

Application Support: It is to be characterized by provision of tailored solutions, encompassing to connected devices potentially of multiple types, and connected with multiple technologies, that are connected to a networks of multiple CSPs (Communication Service Providers).

Solution Provider: It should also be regarded typically as an enabler for a large system development initiative, rather than the standalone offering. These IOT platforms are generally used by systems integrators to support turnkey and the client-specific solutions.

IoT Application Development (5 of 14)

Reference points of IOT service layers

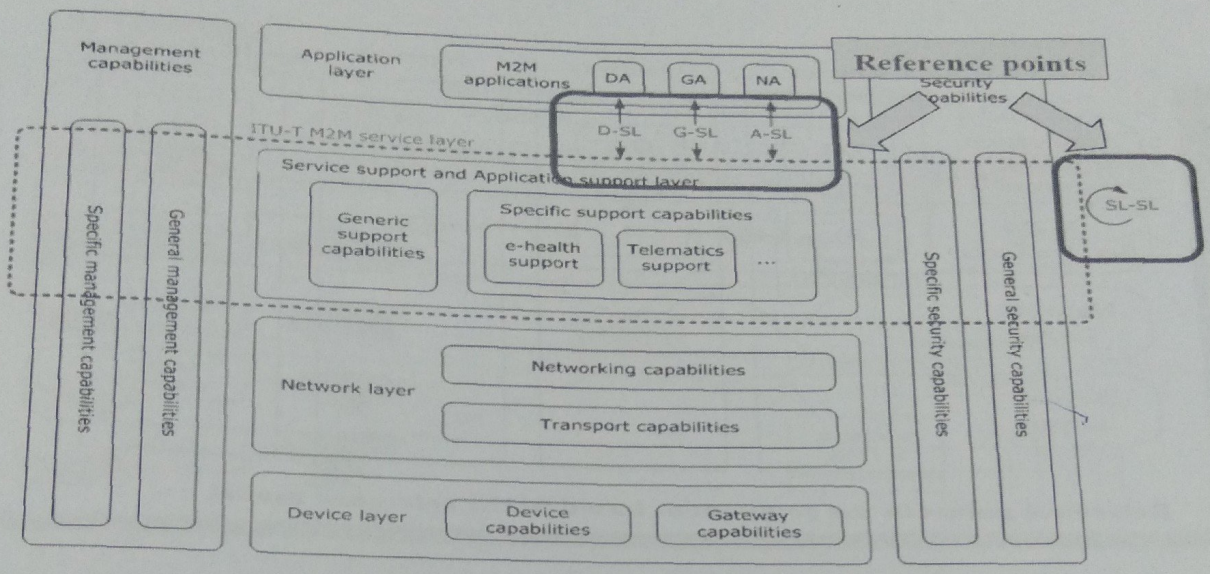


Figure 3-17. IoT Application Development (5 of 14)

Notes:

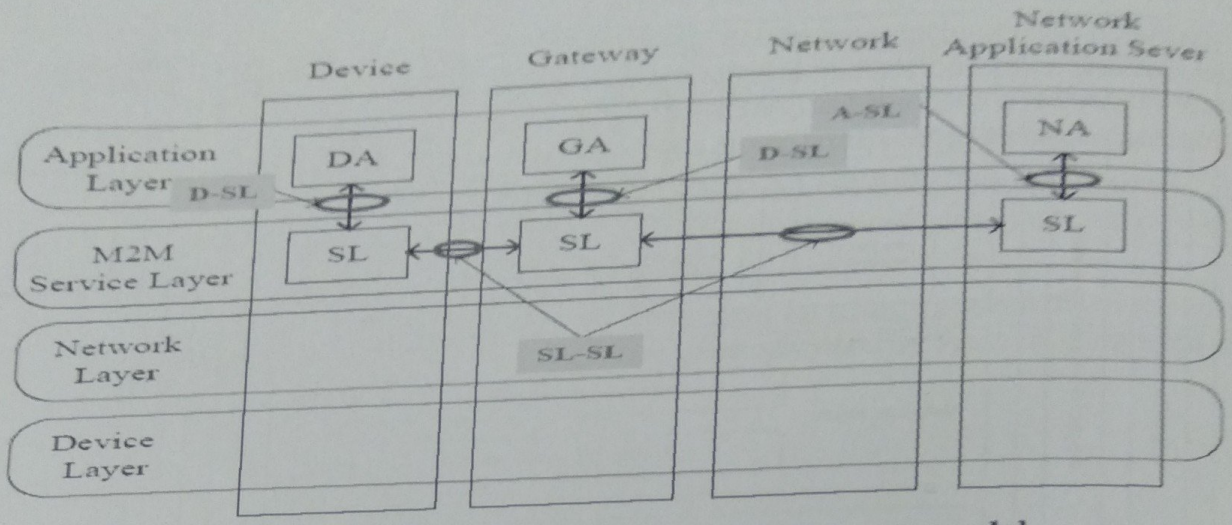
There are three types of IOT applications on top of the IOT service layer: device application (DA), gateway application (GA) and network application (NA). DA, GA and NA reside, respectively, in device, gateway and network application server. All these applications use capabilities provided by the IOT service layer.

Four reference points are identified for the ITU-T IOT service layer: D-SL, G-SL, A-SL and SL-SL.

- D-SL: reference point between DA and IOT service layer
- G-SL: reference point between GA and IOT service layer
- A-SL: reference point between NA and IOT service layer
- SL-SL: reference point between different IOT service layers

IoT Application Development (6 of 14)

IBM ICE (Innovation Centre for Education)



Reference points in the component based M2M reference model

Figure 3-18. IoT Application Development (6 of 14)

Notes:

The four references points are shown in above figure with respect to the component based IOT reference model.

From the figure we can understand that DA and SL function are included in Device, GA and the SL function are included in Gateway, NA and the SL function are included in Network Application Server.

D-SL, G-SL, A-SL, SL-SL, as shown in figure are established as reference points.

It is necessary to clarify requirements of each reference point which can be used by both Generic support capabilities and Specific support capabilities, which reside in the IOT service layer, in order to identify protocols and APIs to be used across these reference points.

IoT Application Development (7 of 14)

IBM

IBM ICE (Innovation Centre for Education)

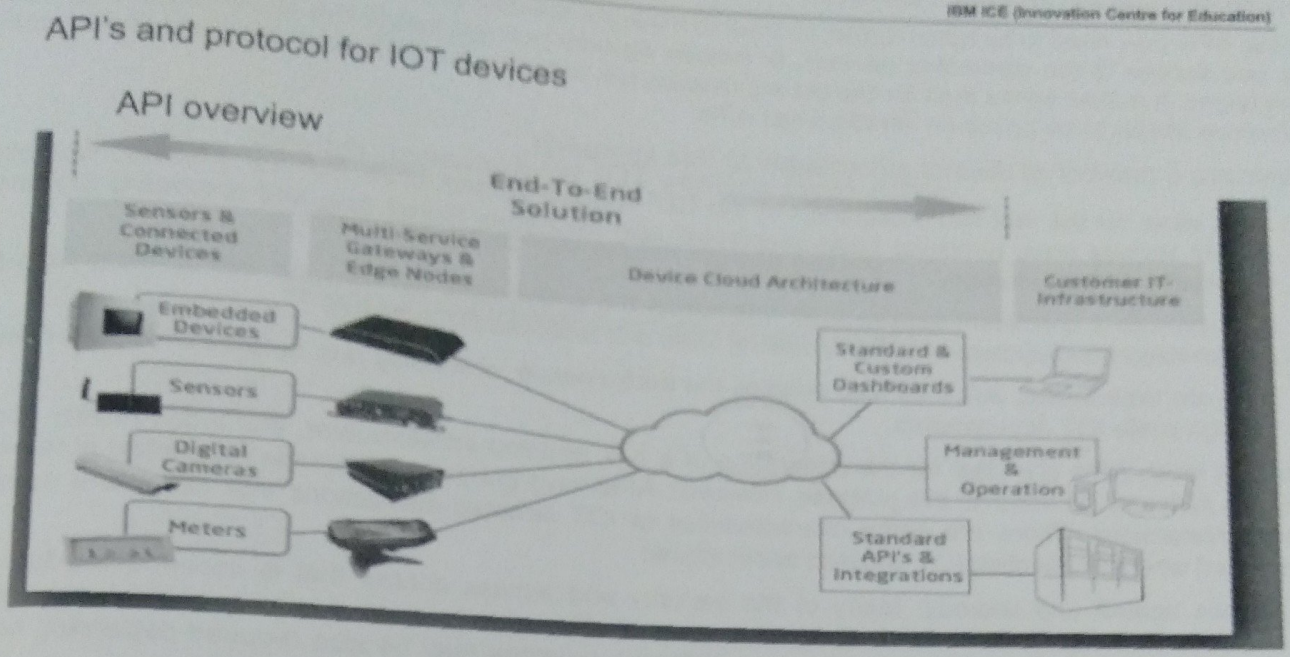


Figure 3-19. IoT Application Development (7 of 14)

Notes:

The API (Application programming interface) is a set of commands, routines, functions, tools, and protocols by which programmers can use when building software applications for a specific operating system. The APIs allow programmers to use the predefined functions and to interact with the operating system, instead of writing the functions from scratch. APIs can be implemented by an application to allow other applications to interact more easily and/or effectively with it. An API expresses a software component in terms of its operations, inputs, outputs, and underlying types by defining functionalities that are independent by their respective implementations, which allows definitions and implementations to vary without compromising the interfaces. And a good API makes it easier to develop a program by providing all the building blocks, which are put together by the programmer.

An API shields applications from the underlying resources, which reduces efforts that are involved in service development. Services intended to be replicated and ported between different execution of environments and hardware platforms. At the same time, services and technology platforms are allowed to evolve independently. A true value of a IOT enabled infrastructure comes not from collecting, integrating and analysing the information from all devices in order to achieve specific business purposes. A quick and effective way to accomplish this practical goal is to connect this IOT enabled infrastructure with the core systems and business processes of the infrastructure. These assets can be made accessible to the IOT enabled infrastructure via APIs. Standardized APIs aim to ensure service interoperability and allow ubiquitous end-to-end service provisioning. Standardized APIs can provide efficiency in scale and, service production, and service development.

The IOT movement represents a significant shift for a number of industries. Connecting and digitizing data from disparate devices is usually disruptive to these industries. However, the foundational elements for the integration of IOT data and application services can be linked. Many companies have already begun to expose APIs that can be used in the context of IOT enabled applications.

These APIs may need to be tuned further in order to minimize the data payloads, adapt the data formats to fit the peculiarities of the connected devices, or include security policies that fit the profile of the data being exchanged. It makes sense then for the communication link between the IOT enabled infrastructure and the enterprise assets to be based on standardized APIs.

There are a number of distinct advantages to this approach:

- APIs allow for the real-time integration of the IOT enabled infrastructure with the e-health information related storage area, removing costs associated with erroneous stored data and respecting regulatory boundaries of data storage.
- APIs provide a consistent approach for integrating the enterprise's services, as well as those from the IOT enabled infrastructure providers, making skills and tools readily available
- APIs are Web-based, and they also enables the performance, scalability and security which are needed for high scale IOT deployments.

At a high level, APIs are ideal for the integration of a IOT enabled infrastructure, and indeed many readily available APIs can be used for this purpose. However, APIs have generally evolved in the context of real-time human interactions. There are some characteristics of IOT enabled interactions that differ, and must be considered when architecting IOT enabled applications:

- Access control and security: Many of the security and access control that is implemented for APIs assumes a human end user with specific permissions. The device-oriented security model is required to ensure appropriate control of the data flow. Different solutions may be also required depending on the access control requirements .
- Synchronicity: Many real-time APIs are synchronous. Many devices in a IOT enabled infrastructure require asynchronous communications for technical and business reasons. Existing APIs may need changes to handle these requirements.
- Scale and bandwidth: IOT enabled communications demand simultaneously high scalability to handle the proliferation of devices, while being constrained bandwidth based on geographical deployment in locations is typical for IT. This requires flexibility in SLAs and optimization of APIs.

IoT Application Development (8 of 14)

IBM

IBM ICE (Innovation Centre for Education)

API's and Protocol for M2M

Design approach for IOT service layer API's

- **Service-oriented architecture (SOA)**
- **Resource-Oriented architecture (ROA)**

Figure 3-20. IoT Application Development (8 of 14) IOT011.0

Notes:

Service-oriented architecture (SOA): Service-oriented architecture is the architectural pattern in the computer software design in which application components provides services to other components via the communications protocol, typically over a network. The principles of the service-orientation are independent of any vendor, product and technology. A service is a self-contained unit of functionality, such as by retrieving an online bank statement. By that definition, a service is an operation that may be discretely invoked. And also in the Web Services Description Language (WSDL), is a service and an interface definition that may list the several discrete services/operations. And if else any where the term service is used for a component that is encapsulated behind the interface. This widespread ambiguity is reflected in what follows the Services can be combined to provide the functionality for a large software application. The SOA makes it easier for software components on computers are connected over a network to cooperate. Every computer can even run any number of services, and each service is built in a way that ensures the service which can exchange that information with any other service in the network without human interaction and without any need to make changes to the underlying program itself.

Resource-Oriented architecture (ROA): Resource-oriented architecture (ROA) is the style of software architecture and programming paradigm for designing and developing software in to the form of resources with "REST full" interfaces. These resources are software components such as (discrete pieces of code and/or data structures) so that can be reused for different purposes. REST (Representational State Transfer) is the approach for getting the information content from a Web site by reading the designated Web page that contains an XML (Extensible Markup Language) file which describes and includes the desired content. The REST approach is basically based on the web technologies such as transfer protocols like HTTP,

identification format as URI (Universal Resource Locator), representation formats such as XML or HTML (Hyper Text Markup Language) and content identifiers as MIME (Multipurpose Internet Mail Extensions) types. REST is neither a product nor a tool: it describes how a distributed software system can be architected. The design of a distributed system receives the stamp of approval of experts in REST, if the design meets a number of constraints: then the system design is called RESTful. REST is consistent with an information publishing approaches many more Web log sites use to describe some aspects of their site content, called RSS (RDF Site Summary). RSS uses RDF (Resource Description Framework), a standard way to describe a Web site or other Internet resources.

IoT Application Development (9 of 14)

API's and Protocol for IOT

Existing API's and protocol for IOT service layer
- Modbus
- UPNP
- IGD
- NAT-PMP
- DPWS
- BITXML
- CoAP
- SOAP
- OAuth
- Web socket

Notes:

A Protocol is a uniform set of rules that enables two devices to connect and transmits data to one another. Protocols determine how data is transmitted between the computing devices, and over networks. Key capabilities of a protocol include the following are type of error checking to be used data compression method (if any), how the sending devices will indicates that it has finished a sage and how the receiving device will indicate that it has received the message.

The following are some of the existing APIs and protocols which can be considered for IOT service layer, but not limited to:

Modbus: Modbus is an application layer of messaging protocol, positioned at level 7 of the OSI model, which also provides client/server communication between the devices are connected on different types of buses or networks. Modbus is a serial communication protocol extensively used in SCADA (Supervisory Control And Data Acquisition) systems to establish a communication between RTU (Remote Terminal Unit) and devices.

UPNP: UPNP technology defines that the architecture for pervasive peer-to-peer network connectivity of the intelligent appliances, wireless devices, and the PCs of all form factors. It is designed to bring easy-to-use, for flexible, and standards-based connectivity to ad-hoc or unmanaged networks that whether in the home, in a small business, or public spaces, or attached to the Internet. UPNP technology provides the distributed, open internet working architecture that leverages TCP/IP and the Web technologies are to enable seamless proximity networking in addition to control and then to transfer data among networked devices.

IGD: Internet Gateway Device (IGD) of Standardized Device Control Protocol is an "edge" interconnect device between the residential Local Area Network (LAN) and also the Wide Area Network (WAN), provides connectivity to the Internet. It is also supported by few NAT routers. It is a common method for automatically configuring port forwarding.

NAT-PMP: NAT Port Mapping Protocol (NAT-PMP) is the protocol for automating a process of the creating Network Address Translation (NAT) port mappings. The NAT-PMP allows the computer in the private network (behind a NAT router) to automatically configure the router to allow parties that are outside the private network to contact itself. NAT-PMP runs over UDP. It essentially automates the process of port by forwarding. Included in the protocol is a method for retrieving the public IP address of a NAT gateway, thus allowing the client to make this public IP address and port number known to peers that may wish to communicate with it.

DPWS: Devices Profile for Web Services (DPWS) defines a minimal set of implementation constraints to enables the secure for Web Service messaging, discovery, description, and eventing on resource-constrained. DPWS is aligned with a Web Services technology and includes numerous extension points allowing for seamless integration of the device-provided services in enterprise-wide application by scenarios. DPWS builds on the following core Web Services standards such as WSDL 1.1, XML Schema, SOAP 1.2, WS-Addressing, and the further comprises WS-Metadata Exchange, WS-Transfer, WS-Policy, WS-Security, WS-Discovery and WS-Eventing.

BiTXML: The BiTXml communication protocol has been designed to implement a presentation level of the (OSI-based) communications stack reference, with the main goal to standardize the way commands and control information are to be exchanged for the specific target of IOT communication demands (i.e. communication with generic devices with or without processing the power on board - like sensors, actuators, as well as air conditioning systems, lifts, etc. or a combination of them).

CoAP: Constrained Application Protocol (CoAP) is a specialized web transfer protocol for the use with constrained networks and nodes for the machine-to-machine applications such as smart energy and for building automation. CoAP provides a method/response interaction model between the application end-points, supports built-in the resource discovery, and includes key web concepts such as URIs and the content-types. CoAP easily translates to HTTP for an integration with the Web while meeting specialized requirements such as multicast support, low overhead and simplicity for constrained environments.

SOAP: Simple Object Access Protocol (SOAP) is the lightweight protocol intended for exchanging the structured information in to the decentralized, distributed environment. It uses XML technologies to define an extensible messaging framework providing a message constructer that can be exchanged over a variety of underlying protocols. The framework has been designed to be an independent of any particular programming model and other implementation specific semantics.

OAuth: The OAuth protocol is a authorization framework enables a third-party application to obtain limited access to an HTTP service, either on behalf of the resource owner by orchestrating an approval interaction between the resources owner and the HTTP service, or by allowing a third-party of the application to obtain access on its own behalf.

WebSocket: The WebSocket protocol enables the two-way communication between clients running untrusted code in a controlled environment to a remote host that has opted-in to the communications from that code. The security model used for this is the origin-based security model is commonly used by web browsers. The protocol consists of an opening handshake followed by basic message framing, layered over the TCP. The goal of this technology is to provide a mechanism for browser-based applications that need two-ways of communication with servers that does not rely on the opening multiple HTTP connections (e.g., using XMLHttpRequest or <iframe>s and long polling)

IoT Application Development (10 of 14)

API's and protocol for IOT

IOT protocol structure and stacks

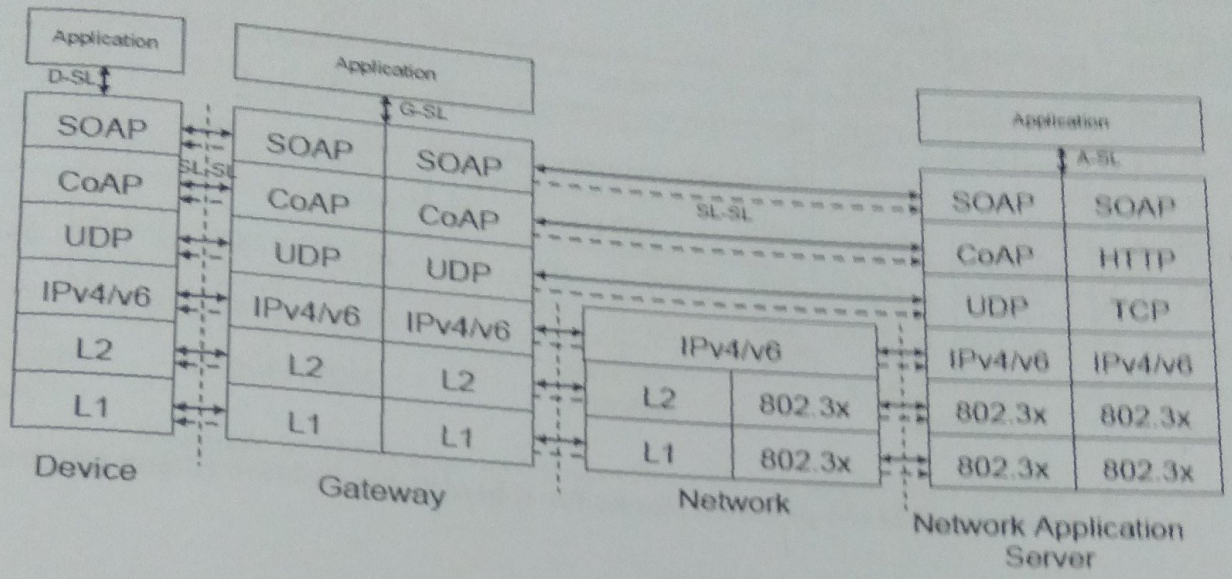


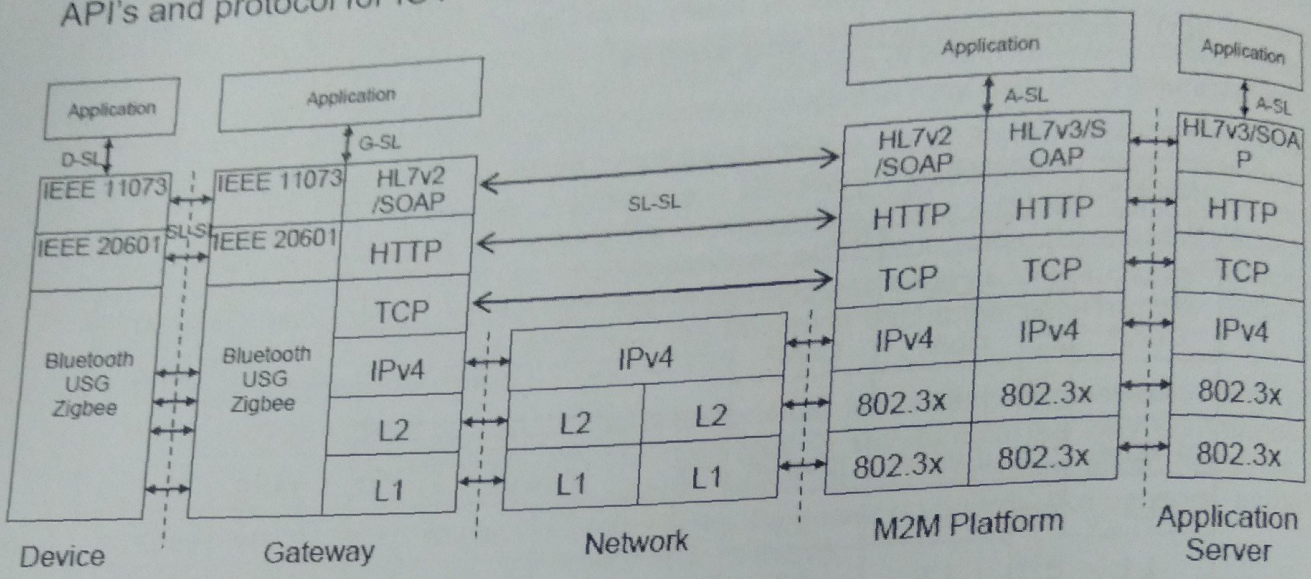
Figure 3-22. IoT Application Development (10 of 14)

Notes:

In order to introduce the IOT service layer related protocol layering and stacks, an example of protocol stacks in the component-based IOT reference model is shown in the above figure. The components of the component-based IOT reference model (device, gateway, IOT platform and application server) are all involved in the application related operations.

There are 2 cases of connection types between devices and IOT platform. The first case is when a device communicates with IOT platform via a gateway(s), shown in the above figure with continuous lines, and the second case is when a device communicates with IOT platform directly without a gateway(s), shown in the above figure with dashed lines.

IoT Application Development (11 of 14)



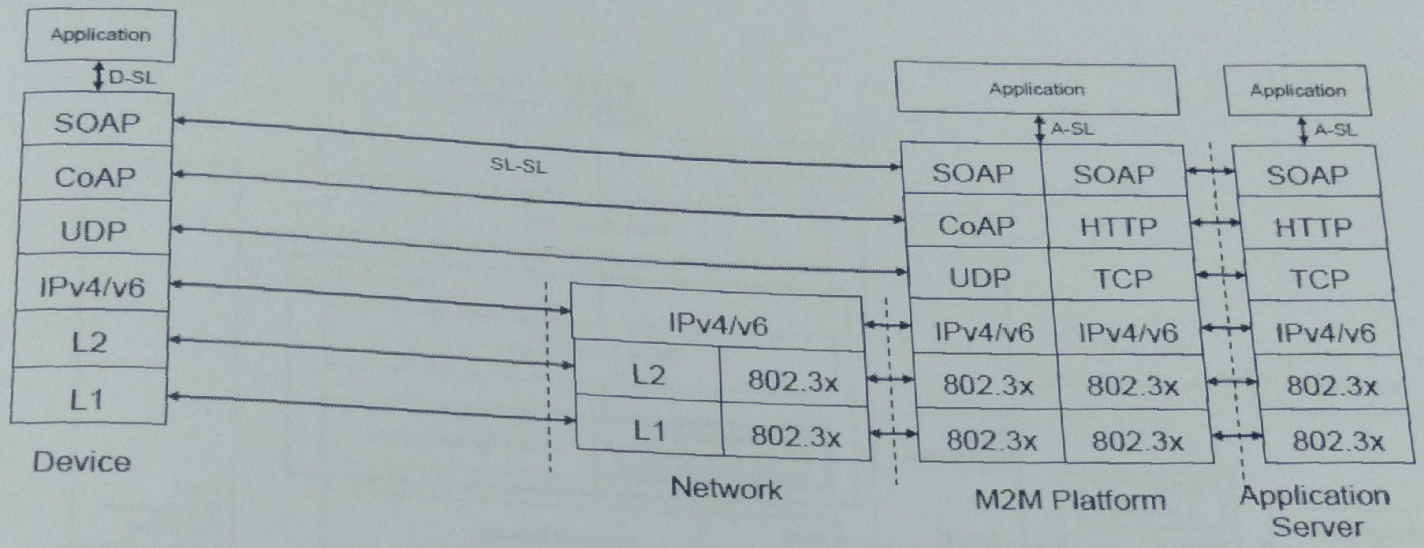
Example of M2M protocol stacks for e-health application (using gateway)

Notes:

Figure above shows a concrete application instance that utilizes the component-based IOT reference model for an e-health application. This concrete instance provides a specific example of IOT protocol stacks where, for example, a scale or a sphygmomanometer is used as device connected to a gateway, and a PHR server is used as application server. This example shows that IEEE11073/IEEE20601 is used for the SL-SL reference point between device and gateway, HL7v2/SOAP/HTTP is used for the G-SL, A-SL and gateway-platform SL-SL reference points , and HL7v3/SOAP/HTTP is used for the A-SL reference point related to NA.

IoT Application Development (12 of 14)

API's and protocol for IOT



Example of M2M protocol stacks for e-health application (without gateway)

Figure 3-24. IoT Application Development (12 of 14)

Notes:

Another concrete application instance in figure 6 shows a specific example of M2M protocol stacks where, for example, a smart phone is used as device connected with the network without a gateway and a PHR server is used as application server. This example shows the usage of SOAP/CoAP for the D-SL and device-M2M platform SL-SL reference points, and SOAP/HTTP for the A-SL reference point related to NA.

IoT Application Development (13 of 14)

Requirements of API's and protocols with respect to the IOT service layer

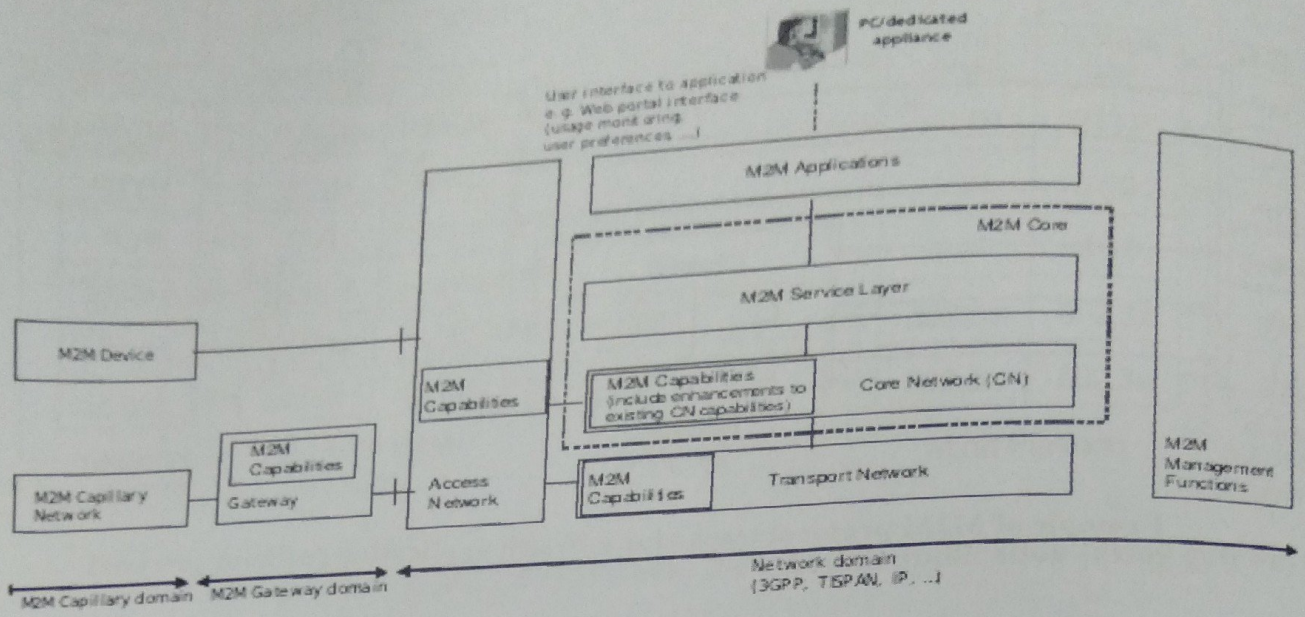


Figure 3-25. IoT Application Development (13 of 14)

Notes:

Main requirements for support of e-health applications: In the case of e-health applications, specific requirements should be considered for each reference point in addition to the common requirements related to generic support capabilities. These additional requirements for each reference point are based on the following e-health specific requirements as described in :

- Security for personal health information
- Privacy protection
- e-health device profile support
- Time synchronization and time stamping
- Audit trail support

Examples of attributes for APIs and protocols: Because of the large number of potential APIs and protocols, their classification and analysis is useful information for developers when selecting suitable protocols for IOT service layer. The following provides some examples of attributes to be considered for APIs and protocols with respect to the various interfaces:

- Interface for device: gateway, device – network application server and device – device: protocol loads (information volume, connectionless/connection-oriented), routing capability, IP based/non IP based;
- IOT platform interface: communication security, scalability, real time capability, multitask capability, stateful/stateless;

- Application server interface: Internet compatibility (with other Internet services);
- Attributes common to all above indicated interfaces: expandability, usability, openness (including open source projects).