Toward a Comprehensive Platform for IoT Applications in Vietnam Using Machine to Machine Communication

Nguyen Tai Hung

Hanoi University of Science and Technology, Hanoi, Vietnam

hung.nguyentai@hust.edu.vn

Abstract *- The Internet of Things (IoT) applications like Home Automation (HA) or Smart Grid presents one of the biggest growth potential in the Machine-to-Machine (M2M) communication today. Thanks to M2M technologies' advances recently, the sensors/actuators and smart meters are expected not only require human intervention in sensing environmental parameters, meter the energy/water consumption level and send them back to central platform for processing but also give out proper & automatic responses to the situation. However, there are still many challenges in designing communication protocols and platform to exchange and share of collected information and other various constrains such as the variety of end devices, real-time nature and extra-low power consumption. This article investigates on a number of existing communication protocols and platforms that can be adopted for M2M application development and deployment in Vietnam conditions.*

Keywords: *M2M; IoT; M2M platform; M2M Protocols; API; Home Automation.*

I. INTRODUCTION

In the most basic sense, M2M is a technology of the future, where a smart device will interact and communicate via a communications network. In order to function, each device must be outfitted with a communication module, and in many cases a sensor, for data collection. With numerous advances in wireless communication and in the base-band processing technologies, these devices can be easily placed in diverse locations, far away from one another, or form a central system through various access network technologies, like xDSL, Satellite, GPRS, EDGE, UMTS or Wi-Fi. In general M2M communication often refers to a system of remote sensors, middleware, software and applications that are continuously transmitting data to a central system. The main goal of M2M communications is to enable sharing of information between electronic systems autonomously. However, despite its real-time application and lots of benefits, research in M2M communication is still in its infancy, and faces many technical challenges, including system architecture, specifications of the platform, energy efficiency, cost effectiveness, reliability, privacy, and security [17]. This paper is about to address some of those issues, focusing on the communication protocols and unified platform and service exposure capability of M2M communication. The paper contributions are the evaluation of some of the potential candidates for communication protocols on a test-bed that built around the in-house and open source-based M2M platform. The paper also shows the usefulness of the platform API (Application Programing Interface) with two demonstrated applications, named Home Automation and Smart Meter.

II. PREVIOUS WORK

To our best of knowledge, in the field of M2M communication and IoT most of the researches until recently are focusing on the sensing technologies and coupled wireless and wired local networking technologies like RFID or IEEE 802.15.4 and there are a few of researching works on the M2M platform, though, worthy to analyze here. Regarding to the M2M platform architecture, amongst others, the most significant works are inside working groups of the standardization body ETSI and industrial forum OneM2M.

An ETSI Technical Committee [21,1] is developing standards for M2M Communication. The aim of this group is to provide an end-to-end view of M2M standardization cooperating with ETSI's activities on Next Generation Networks and 3GPP standards initiative for mobile communication technologies and Section III.A below will describe the details of M2M communication architecture defined by ETSI. There is another important standardization effort, oneM2M [5], has been setting up by the collaboration of international standard bodies such as ETSI, ATIS, TIA, etc. as well as equipment manufacturers all over the world. The purpose of this oneM2M forum is to develop technical specifications for addressing the need of a common M2M Service Layer that can be readily embedded within various hardware and software, and relied upon to connect the huge amount of devices in the field with M2M application servers worldwide. A critical objective of oneM2M is to attract and actively involve organizations from M2M-related business domains such as: telematics and intelligent transportation, healthcare, utilities, industrial automation, smart homes, etc.

The work in [3] explored M2M communication applications and scenarios, which are growing and leading the way to new business cases. The work revealed the practical requirements and threats of M2M application scenarios and point out two main aspects, namely the unpredictable connectivity to the core network and the demand for high configurability and flexibility of M2M devices. While this work attempted to identify security threats against M2M communications, the exact technologies upon which M2M communications are based were not taken into account. On the other hand, various challenges in designing home M2M network are presented in [4]. This work shows that the home networks are expected to require effective M2M gateways to facilitate communication among the various M2M devices and to provide a connection to a backhaul (e.g., with the core communication network of SG). While the backhaul connection may be fiber, cable, DSL, Ethernet, or even cellular, the authors suggest that it is also important to choose appropriate network protocols to enable M2M devices to communicate inside a home environment.

And the conclusion here is that there are very few researches, in the past, who have tried to investigate the details of a comprehensive platform in M2M communication which hosts various mechanisms from perspective of the service abstraction, service composition, service management and device/sensor management. This is the reason why we wrote this paper to provide the readers with an overview concept of what M2M communication platform is. The paper also provides the inside look in to the components of the M2M platform, like the middleware, the protocols and the service abstraction (API) layer. Finally a prototype implementation of the said platform and its application using open-sources are also presented on the paper.

Thus, the remainder of this paper reviews, in section III, the important concepts of M2M communication and more importantly investigating the various options for protocols to be used in M2M communication while section IV provides the details of our implementation of so-called HUST M2M platform with a sample application called home automation. We conclude with lessons learned and future works.

III. DETAILS ABOUT M2M COMMUNICATIONS

This section summarizes the basic concepts and mechanisms using in M2M communication. Sub-section A shows the reference architecture with all of functional blocks and components comprising of the complete end to end M2M system, sub-section B presents the potential protocols could be used in M2M for components to communicate and share information with each other and with central system, while sub-section C describes the most important element: the central platform as well as its API helping to develop a wide range of real-life applications.

III. 1. M2M Communication Architecture

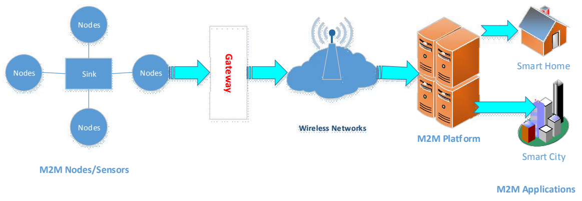


Figure 1: ETSI M2M Architecture

High–level M2M architecture provides an overview of the components of a system and the relationship between the individual components. It provides the starting point for a stepwise approach to the description of the functional architecture. Although every particular deployment of M2M is unique, there are four basic stages that are common to most M2M based applications [2]: Collection of data, Transmission of data through a communication network, Assessment of data, and Response to the available information. ETSI TC M2M has adopted the standard architecture (Figure 1) which allows for a common understanding of such high-level system that is under standardization. This high level architecture fully endorses the need for M2M service capabilities that are exposed towards applications, whether it is in the network, in the device or in the gateway. According to definition, the architecture of M2M Communication System is composed of three domains: Node domain, Network domain and Application domain. M2M Node domain is used to sense the activity under area of interest and collect the data for further processing [2].

III. 2 M2M Communication Protocols

As mentioned above, there are a lot of types of the devices (from tiny sensors, actuators to the gateway and meters) using in various applications of the M2M systems, some of them connect with the integrated communication stack while others not. That reality makes the choice of communication protocols for an M2M network/application harder to be standardized and optimized. Furthermore, the nature of ultra-low power consumption of those devices make the task even harder that chosen protocol need to be powerful enough to accomplish the application requirements while have to be light for saving power (at the device). Thus, our main motivation here was to create an M2M test-bed in our university facility from where to test different candidates for using as communications protocols and for innovative applications that could be applied to a wide range of real-life scenarios. It is expected to provide a brief yet accurate description of the key protocols that can be used for implementing the M2M Communication. Our own study, which are on-going work in our test-bed, lead to following list of potential protocols could be used alternatively or jointly to solve different needs of the communication between machines: ***CoAP*** *(Constrained Application Protocol);* ***MQTT*** *(Message Queue Telemetry Transport);* ***XMPP*** *(Extensible Messaging and Presence Protocol);* ***RESTFUL Services*** *(Representational State Transfer);* ***AMQP*** *(Advanced Message Queuing Protocol); and* ***Web-sockets****.*

CoAP is a synchronous request/response application layer protocol that was designed by the Internet Engineering Task Force (IETF) to target constrained-recourse devices. It was designed by using a subset of the HTTP methods making it interoperable with HTTP [10]. CoAP runs over UDP to keep the overall implementation lightweight. It uses the HTTP commands GET, POST, PUT, and DELETE to provide resource-oriented interactions in a client-server architecture. CoAP is a request/response protocol that utilizes both synchronous and asynchronous responses. The reason for designing a UDP-based application layer protocol to manage the resources is to remove the TCP overhead and reduce bandwidth requirements. Additionally, CoAP supports unicast as well as multicast, as opposed to TCP, which is by its nature not multicast-oriented. Running on the unreliable UDP, CoAP integrated its own mechanisms for achieving reliability. Two bits in the header of each packet state the type of message and the required Quality of Service (QoS) level. There are 4 message types:

1. *Confirmable*: A request message that requires an acknowledgement (ACK). The response can be sent either synchronously (within the ACK) or if it needs more computational time, it can be sent asynchronously with a separate message.

2. *Non-Confirmable*: A message that does not need to be acknowledged.

3. *Acknowledgment*: It confirms the reception of a confirmable message.

4. *Reset*: It confirms the reception of a message that could not be processed.

MQTT was released by IBM and targets lightweight M2M communications. It is an asynchronous publish/subscribe protocol that runs on top of the TCP stack. Publish/subscribe protocols meet better the M2M communication requirements than request/response since clients do not have to request updates thus, the network bandwidth is decreasing and the need for using computational resources is dropping. In MQTT there is a broker (server) [12] that contains topics. Each client can be a publisher that sends information to the broker at a specific topic or/and a subscriber that receives automatic messages every time there is a new update in a topic he is subscribed.

The MQTT protocol is designed to use bandwidth and battery usage sparingly, which is why, for example, it is currently used by Facebook Messenger. MQTT ensures reliability by providing the option of three QoS levels:

1. *Fire and forget*: A message is sent once and no acknowledgement is required.
2. *Delivered at least once*: A message is sent at least once and an acknowledgement is required.
3. *Delivered exactly once*: A four-way handshake mechanism is used to ensure the message is delivered exactly one time.

XMPP was designed for chatting and message exchanging. It was standardized by the IETF over a decade ago, thus being a well-proven protocol that has been used widely all over the Internet. However, being an old protocol, it falls short to provide the required services for some of the new arising data applications. For this reason, last year, Google stopped supporting the XMPP standard due to the lack of worldwide support. However, lately XMPP has re-gained a lot of attention as a communication protocol suitable for the M2M Communication. XMPP also runs over TCP and provides publish/subscribe (asynchronous) and also request/response (synchronous) messaging systems. It is designed for near real-time communications and thus, it supports small message footprint and low latency message exchange [14]. As the name explicitly states, XMPP is extensible and allows the specification of XMPP Extension Protocols (XEP) that increase its functionality. XMPP has TLS/SSL security built in the core of the specification. However, it does not provide QoS options that make it impractical for M2M communications. Only the inherited mechanisms of TCP ensure reliability. XMPP supports the publish/subscribe architecture that is more suitable for the M2M Communication in contrast to CoAPs request/response approach. However, XMPP uses XML messages (eXtensible Markup Language) that create additional overhead due to unnecessary tags and require XML parsing that needs additional computational ability which increases power consumption.

RESTFUL SERVICES is not really a protocol but an architectural style. It was first introduced by Roy Fielding in 2000 [15], and it is being widely used ever since. REST uses the HTTP methods GET, POST, PUT, and DELETE to provide a resource-oriented messaging system where all actions can be performed simply by using the synchronous request/response HTTP commands. It uses the built-in accept header of HTTP to indicate the format of the data that it contains. The content type can be XML or JSON (JavaScript Object Notation) and depends on the HTTP server and its configuration. REST is already an important part of the M2M Communication because it is supported by all the commercial M2M cloud platforms. Moreover it can be implemented in smartphone and tablet applications easily because it only requires an HTTP library which is available for all the Operative Systems (OS) distributions.

AMQP is a protocol that arose from the financial industry. It can utilize different transport protocols but it assumes an underlying reliable transport protocol such as TCP [16]. AMQP provides asynchronous publish/subscribe communication with messaging. Its main advantage is its store-and-forward feature that ensures reliability even after network disruptions. It ensures reliability with the following message-delivery guarantees [16]:

1. *At most once*: means that a message is sent once either if it is delivered or not.
2. *At least once*: means that a message will be definitely delivered one time, possibly more.
3. *Exactly once*: means that a message will be delivered only one time.

Security is handled with the use of the TLS/SSL protocols over TCP. Recent research has shown that AMQP has low success rate at low bandwidths, but it increases as bandwidth increases. Another study shows that comparing AMQP with the previously mentioned REST, AMQP can send a larger amount of messages per second [17]. Additionally, it has been reported that an AMQP environment with 2,000 users spread across five continents can process 300 million messages per day [17].

WEB-SOCKET protocol was developed as part of the HTML 5 initiative to facilitate communications channels over TCP. Web-socket is neither a request/response nor a publish/subscribe protocol. In Web-socket a client initializes a handshake with a server to establish a Web-socket session. The handshake itself is similar to HTTP so that web servers can handle Web-socket sessions as well as HTTP connections through the same port [18]. However, what comes after the handshake does not conform to the HTTP rules. In fact, during a session, the HTTP headers are removed and clients and servers can exchange messages in an asynchronous full-duplex connection. The session can be terminated when it is no longer needed from either the server or the client side. Web-socket was created to reduce the Internet communication overhead while providing real-time full-duplex communications.

III. 3. M2M Platform and API

Though, there has been a significant activity in the open source community relating to the design and development of platforms and operating environments upon which M2M applications can be built, this section tries to figure out the most important aspects on the design of the platforms used to realize M2M communication. In our opinion, a platform designed for M2M applications must cover to the needs of four different types of users namely *a) End Users of M2M applications, b) Application Developers, c) Sensor Providers and d) Administrators who operate and maintain the platform*. Thus, the most important thing, a platform considered to be complete if it consists of a set of services, typically delivered over a Wide Area Network or the Internet using IP based transports such a HTTP, TCP or UDP and fully integrated with application-level protocols as investigated on sub-section B above. Below are some discussions about different aspects of a so-called complete platform for M2M network that we think need to be thoroughly investigated while trying to building such system.

Platform Users

There are various types of users of an M2M platform which are described subsequently. First category of those users are application developers who register themselves to the M2M platform and use the platform services to develop and deploy applications. The applications themselves may be distributed onto both "edge" devices such as sensor nodes or gateway devices and back-end cloud hosted servers. Application developers are provided necessary computing resources and Application Programming Interfaces (APIs) in order to develop applications and then deploy them on the platform. Application developers use API keys to identify and authenticate themselves to the platform and get access to services and resources as per their entitlement. Sensor providers are those who own and/or operate sensors and contribute sensor observations to the platform, either for their own private use or for use by others based on access control and privacy policies. Sensor providers use APIs provided by platform to push data to the platform. Platform Administrators use the services, APIs and tools provided by the platform to manage and monitor users, services & devices. Administrators provide compute, storage and network resources to application developers and keep track of resource usage. Finally we have end users of M2M applications who has huge number and consumes the applications developed by application developers.

Platform Services

There are different types of services need to be provisioned on the platform. They include services data management, sensor/device management, data storage, analytics & visualization and application & user management.

- ***Device Management Services*** are used by both application developers and administrators to register sensors and gateway devices, give them a unique identity, address the devices, check their health and connectivity status, install and update software on the devices and access resources and consume services from the devices.

- ***Sensor Services*** are used by M2M applications to register sensors/observers in the platform, create meta-data about the sensors. The metadata includes the features of the sensors and the real-world phenomena that they measure or observe, the geographical location of the sensors and the real-world entity observed. Sensor Services are also used to capture, store and query sensor observations. Sensor Services are therefore at the core of any M2M platform..

- ***Storage Services*** are used by application developers to store application specific data persistently. This data is typically not the sensor observation data, since that is taken care by the Sensor Services. Typically Storage Services include feature of Analytics Services that need to be highly scalable. Complexities of the underlying infrastructure need to be hidden from the application developers and convenient APIs must be designed to enable applications to easily access the analytics platform. Moreover, it must be realized that, analytics is very problematic and only valid within domain. Hence the actual analytics algorithms are developed as part of the applications themselves. These algorithms may be written in a variety of languages. Often scripting languages such as Python are used for the actual algorithms. The platform must therefore be multiple-languages support.

- ***Visualization*** of sensor data is another important requirement and goes hand in hand with analytics. The M2M platform must provide necessary tools and APIs for creating rich visualization of captured sensor data as well as processed data resulting from analytics.

- ***Application and User Management*** services are used by platform administrators to create tenants on the M2M platform, provide resources and provision platform services to tenants. Application developers themselves use these services to request for resources and services and manage the life cycle of the applications.

Platform Architecture

Normally the software platform architectures for M2M Communication in the literature incorporates a middleware as an abstraction layer and follows a Service Oriented Architecture (SOA) approach in the next generation Internet. The adoption of the SOA principles allows decomposition of singular systems into applications consisting of a set of simpler and well-defined components. The use of common interfaces and standard protocols allows for flexibility in service composition and reduction of the time necessary to adapt to the changes imposed by the application evolution. Since there are no commonly accepted layer architectures for M2M, there is a difficulty in specifying a common set of services and an environment for service design and composition. A typical integrated architectural approach often proposes following M2M specific layers over the SOA namely: *a) Service Composition, b) Service Management and c) Object Abstraction*.

Service Composition provides for building specific applications through the composition of atomic services derived from various connected objects. *Service management* includes: object dynamic discovery, status monitoring, and service configuration. This basic set could be extended with additional functionalities involving QoS management, semantic interoperability functions etc. *Object Abstraction* is needed for heterogeneous connected objects for harmonizing the access to the different objects/devices. Two types of service-oriented architectures stand out as potential candidates to enable uniform interfaces to smart objects: the Representational State Transfer (REST) [19] and WS-* [20] Web services.

Platform API

This sub-section describes an Open API specifications [21], for an M2M platform, from ETSI. By our opinion it is probably the most valuable achievement at this moment (in the standardization process of M2M Communication technologies). Actually, in this Open API we can see the big influence of Parlay specification. Parlay Group leads the standard, so called Parlay/OSA API, to open up the networks by defining, establishing, and supporting a common industry-standard APIs. Parlay Group also specifies the Parlay Web services API, also known as Parlay X API, which is much simpler than Parlay/OSA API to enable IT developers to use it without network expertise [22]. The goals are obvious, and they are probably the same as for any unified API. One of the main challenges in order to support easy development of M2M services and applications will be to make M2M network protocols "transparent" to applications.

Providing standard interfaces to service and application providers in a network independent way will allow service portability. At the same time an application could provide services via different M2M networks using different technologies as long as the same API is supported and used. This way an API shields applications from the underlying technologies, and reduces efforts involved in service development. Services may be replicated and ported between different execution environments and hardware platforms. A standard open M2M API with network support will ensure service interoperability and allow ubiquitous end-to end service provisioning.

The Open API relates to several interfaces of M2M architecture (Figure 2). For example:

- The interface between the platform and external service providers running their services remotely,
- The interface between the platform and the customer applying the features offered by the platform,
- A set of interfaces supporting additional functionality (installation support, access to remote databases, remote operation and management of platform), etc.

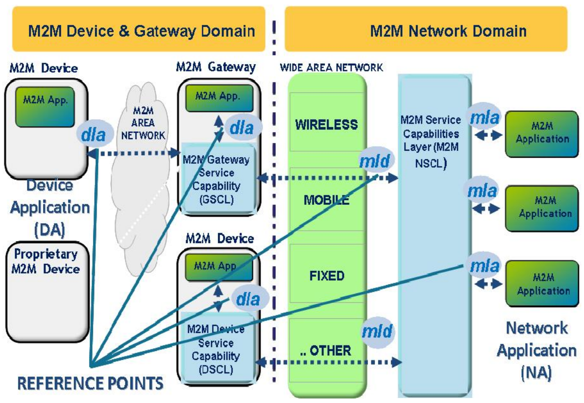


Figure 2. M2M Interfaces from ETSI [1]

Main API sections for Services Capabilities Level are:

- Subscription and Notification (e.g. Publish/Subscribe).
- Grouping and Transactions.
- Application Interaction: Read, Do, Observe.
- Compensation (micro-payment).
- Sessions.

Let us provide more details for Open API categories and make some comments:

Grouping. A group here is defined as a common set of attributes (data elements) shared between member elements. On practice it is about the definition of addressable and exchangeable data sets. Just note, as it is important for our future suggestions, there are no persistence mechanisms for groups *Transactions*. Service capability features and their service primitives optionally include a transaction ID in order to allow relevant service capabilities to be part of a transaction. Just for the deploying transactions and presenting some sequences of operations as atomic. In the terms of transactions management Open API presents the classical 2-phase commit model. By the way, we should note here that this model practically does not work in the large-scale web applications. We think scalability is very important because without it we cannot have "billions of connected devices".

Application Interaction part is added in order to support development of simple M2M applications with only minor application specific data definitions: readings, observations and commands. Application interactions build on the generic messaging and transaction functionality offer capabilities considered sufficient for most simple application domains.

Messaging. The Message service capability feature offers message delivery with no message duplication. Messages may be unconfirmed, confirmed or transaction controlled. The message modes supported are single Object messaging, Object group messaging, and any object messaging; (it can also be Selective object messaging). Think about this as Message Broker.

Event notification and presence. The notification service capability feature is more generic than handling only presence. It could give notifications on an object entering or leaving a specific group, reaching a certain location area, sensor readings outside a predefined band, an alarm, etc. It is a generic form. So, for example, geo fencing should fall into this category too. The subscriber subscribes for events happening at the Target at a Registrar. The Registrar and the Target might be the same object. This configuration offers a publish/subscribe mechanism with no central point of failure.

Compensation. Fair and flexible compensation schemes between cooperating and competing parties are required to correlate resource consumption and cost, e.g. in order to avoid anomalous resource consumption and blocking of incentives for investments. The defined capability feature for micro-payment additionally allows charging for consumed network resources. It is very similar, by the way, to Parlay's offering for Charging API. Again it is a big question from the modern large-scale applications point of view: shall we develop a special API for the compensations or create a rich logging functionality where the external log processing should be responsible for the things as charging.

Sessions. In the context of Open API a session shall be understood to represent the state of active communication between Connected Objects. Open API is REST based, so, the endpoints should be presented as some URI's capable to accept (in this implementation) the basic commands GET, POST, PUT, DELETE. Actually, ETSI uses the Smart Meter profile as 'proof of concept' for the M2M service platform in Release 1 […].

IV. HOME AUTOMATION APPLICATION

At HUST, we started a research program in early 2015 to build a unified test-bed (Figure 3) for M2M Communication. And in that project, our team has been able to develop, amongst other things, a light-weight M2M platform utilizing open source and with an RESTful-based API. With that API, various applications of real life could be developed, tested and benchmarked before going in to the real deployment. But for this testbed we specifically try to prototype three kinds of application that are useful and feasible for commercial deployment in Vietnam which we called Smart Meter, Home Automation (HA) and Location Based Services (LBS). The Smart Meter application provides functions for automatically collecting and metering the usage level of electrical users and send back to back-end server for storage, analyzing and billing while the HA application provides functions for smart home like sensing the temperature, light, humidity, smoke, etc., send back to back end server and may conduct the suitable control tasks sending from central server. Finally the LBS application providing functions of automatically determine the geographically location of end users (mobile phone, car, truck, etc.) and send back to back-end server which will triggered specific service logics to be implemented depend on the location of end users. Subsequent sections present the design of the HUST M2M platform as well as detailed description of the HA application.

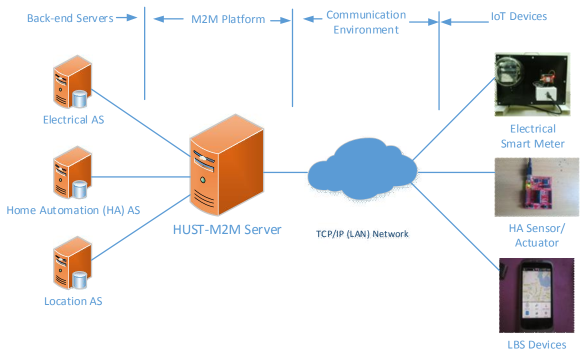


Figure 3. HUST M2M Test-bed

IV. 1. HUST M2M Platform

There are different technical architectural approaches for M2M systems such as e.g., end-to-end Internet approach [3,4], and M2M gateway based approach [5,6]. It is possible to establish an Internet connection from an Internet node to the M2M asset device without any additional intermediate node making transformation into the messages in the end-to-end Internet based approach, if there is at least tiny IP stack also in small embedded devices. This is possible to be done even for such small devices by relying power efficient physical layer and IETF IPv6 Low Power wireless Area Networks (6LowPAN) adaptation layer enabling universal Internet connectivity, the IETF Routing Over Low power and Lossy networks (ROLL) routing protocol enabling availability, and IETF Constrained Application Protocol (CoAP) enabling seamless transport and support of Internet applications [7–15]. However, the challenge is that also the embedded devices which are not IP capable would be required to connect into the Internet. M2M gateway based approach may enable also their connectivity, however, the challenge may be dynamic behavior of wireless systems and need to adapt with different kinds of service back-end systems. Our investigation found out two aspects that most existing M2M platforms are either do not support or support incompletely. The first aspect is the issue of supporting of application-level communication protocols and the second aspect is the way new type of devices (sensors/actuators, meters, gateway, etc.) to be integrated and communicated. On the first aspect, most of platforms just support one or few protocol listed on section III-B while on the device management aspect, most of the platform get difficulty when adding new type of sensors/actuators or meters that doesn't complied with current protocol and/or communication environment. That's why in our research project, we have tried to develop a modular software architecture of the M2M platform that allows easily adding more protocols and device management modules in form of the plug-in. The proposed platform also follows the ETSI standards in which there's a RESTful-based API allowing to develop more application/services without changing or modification of the core parts. Figure 4 illustrates this idea of the designing of platform, and besides the above mentioned features/modules, the platform also provides a flexible SCL [25] that can be deployed in an M2M network, a gateway, or a device. An SCL is composed of small tightly coupled plugins, each one, offering specific functionalities. A plugin can be remotely installed, started, stopped, updated, and uninstalled without requiring a reboot. It can also detect the addition or the removal of services via the service registry and adapt accordingly facilitating the SCL extension.

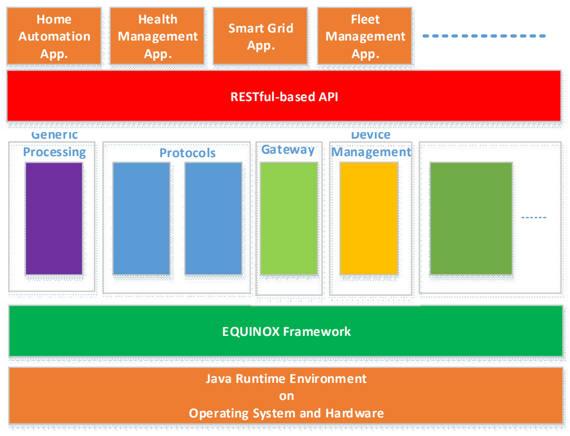


Figure 4. HUST M2M platform functional architecture

The MAIN module provides a protocol-independent service for handling REST request. Specific communication mapping plugins can be added to support multiple protocol bindings such as HTTP and CoAP. The platform can be extended with specific device management mapping plugins to perform device firmware updates by reusing existing protocols such as OMA-DM [24]. It can be also extended by various interworking proxy plugins to enable seamless communication with legacy devices such as ZigBee and Wi-Fi technologies. A new plugin based on the autonomic computing paradigm is designed to enhance HUST M2M resource discovery and self-configuration.

Figure 5 shows the *CORE* plugin implements the *SCL_Service* interface to handle generic RESTful request. It receives a protocol-independent request indication and answers with a protocol-independent response confirm. The *Router* defines a single route to handle every request in a resource controller simply using request URI and method. The *Resource_Controller* implements CRUD methods (Create, Retrieve, Update, and Delete) for each resource. It performs required checking operations such as access right authorization, and resource syntax verification. The *Resource_DAO* provides an abstract interface to encapsulate all access to resource persistent storage without exposing details of the database.

The *Event_Notifier s*ends notifications to all interested subscribers when a resource is created, updated or deleted. It performs filtering operations to discard events not of interest to a subscriber. The *Resource_Announcer a*nnounces a resource to a remote SCL to make it more visible and accessible to other machines. It also handles resource de-announcement. The *Request_Sender* holds discovered protocol-specific clients implementing the *Client_Service* interface. It acts as a proxy to send a generic request via the correct communication protocol. The *Interworking_Proxy* holds discovered interworking proxy units (IPUs) implementing the *IPU_Service* interface, and acts as a proxy to call the correct IPU controller. *Device_Manager* holds discovered Remote Entity Managers (REMs) implementing the *REM_Service* interface, and acts as a proxy to call the correct device manager controller.

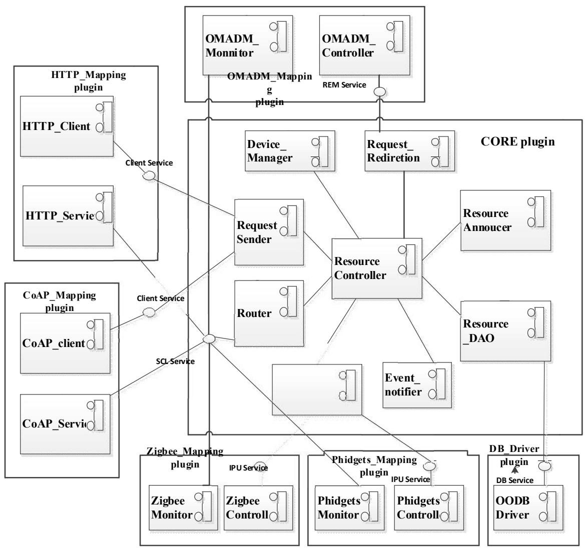


Figure 5. The software modules of HUST M2M platform

The *HTTP_Mapping* plugin provides a bidirectional binding to the HTTP protocol. The *HTTP_Servlet* receives and converts an HTTP request into a generic one, and call the *CORE* plugin via the *SCL_Service* interface. The *HTTP_Client* implements the *Client_Service* interface to send a generic request via HTTP. The *Phidgets_Mapping* plugin provides a bidirectional mapping to interwork with legacy Phidgets devices. The *Phidgets_Monitor* discovers Phidgets devices, and calls the *CORE* plugin to create required resources on the SCL. The *Phidgets_Controller* implements the *IPU_Service* interface to seamlessly perform a generic request via the Phidgets API. The *OMADM_Mapping* plugin provides a bidirectional mapping to manage OMA-DM enabled devices. The *OMADM_Monitor* listens to OMA-DM enabled devices, and calls the *CORE* plugin to create required resources on the SCL. The *OMADM_Controller* implements the *REM_Service* interface to converts generic request into OMA-DM management session. The *DB_Driver* plugin provides an Object Oriented Data Base (OODB) accessible via the *DB_Service* interface. Other plugins can be deployed using the same approach to interwork with additional protocols or to integrate new capabilities.

IV. 2. Home Automation Application

To illustrate the effectiveness of using the platform API in specific application development, this section presents the process of developing the home automation application with the typical temperature auto-sensing function.

Figure 6 depicts the whole application scenario with different components such as sensors and smart meters for acquiring the room condition (temperatures, humidity level, smokes, and so on) or health parameters or the usage of electricity/water. The heart of the whole system is the HUST M2M platform which store and analyze the data collected from those sensors/meters and subsequently give out proper responses/alerts. Besides, the system include the features to allow administrator or user to configure and change the various application parameters and/or purposes.

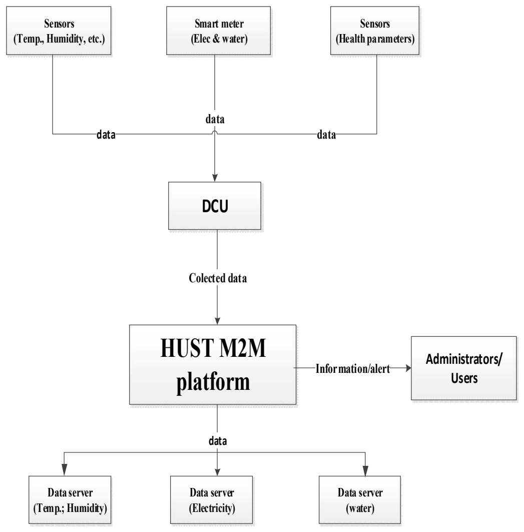

Sensors (Temp., Humidity, etc.)
Smart meter (Elec & water)
Sensors (Health parameters)
data
data
data
DCU
Colected data
HUST M2M platform
Information/alert
Administrators/ Users
data
Data server (Temp.; Humidity)
Data server (Electricity)
Data server (water)

Figure 6: Home Automation Application architecture

However, on this paper we only focus on a feature that allows the system to sense the temperature level and give out the proper responses/alerts as an example of the great potential of using M2M platform to meet various demands of daily people and society life. Thus sensors which are installed in all the home and rooms will send the measured temperature parameters back to application hosted on HUST M2M platform, the application then compare the collected parameters with pre-set threshold levels and subsequently send out the alert or normal condition indicator depend on the comparison results. The app also allow to automatically start the fire-fighter systems in case of the serious alert. Similar features can be applied to other fields like monitoring health condition for old people or monitoring other environment condition. Figure 7 illustrates the work flow of the application while figure 8 describes the details of algorithm for processing the collected data in which the measured data from sensors will be compared with the configurable thresholds, if it is under the thresh old then the LED will be on green to indicate the normal condition, otherwise the LED will turn RED meaning the alarm condition which will lead to the initiation of alarm system as well as the security system to take action (i.e. shutdown) against all the electrical equipment's in the room after a short interval of time (for saving the on-going works). The management procedure of the devices in a house is also provided and presented on figure 9 with combination of several steps inclusion of registration of devices into the application database. For the registered devices (Figure 10), the administrator/user can perform activities like initiation, changing the power level, disable or even shut it down.

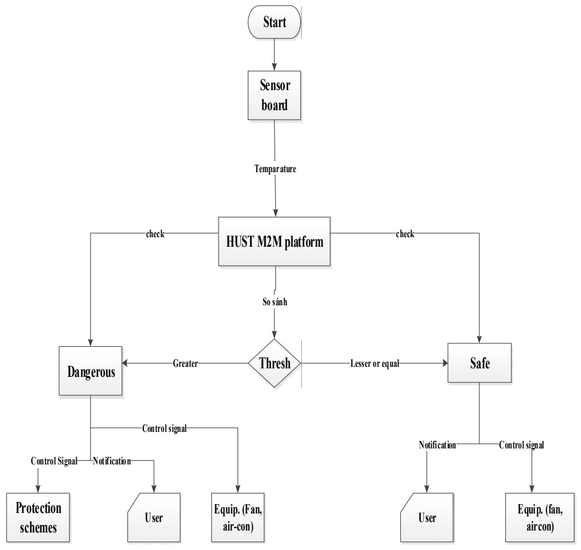


Figure 7: The app working procedure

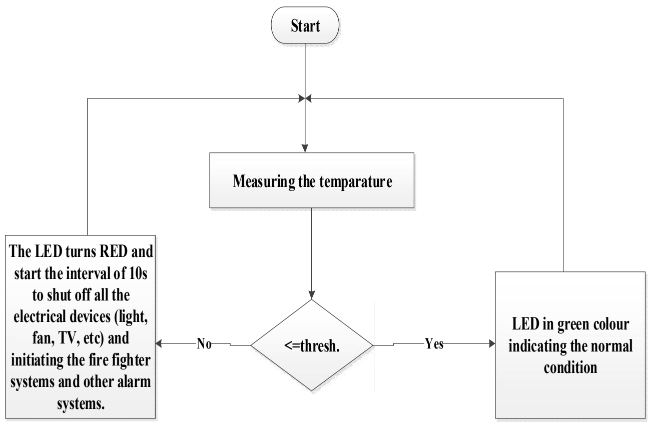

Start
Measuring the temparature
<=thresh.
No
Yes
The LED turns RED and start the interval of 10s to shut off all the electrical devices (light, fan, TV, etc) and initiating the fire fighter systems and other alarm systems.
LED in green colour indicating the normal condition

Figure 8. Processing algorithm for collected data (temp.)

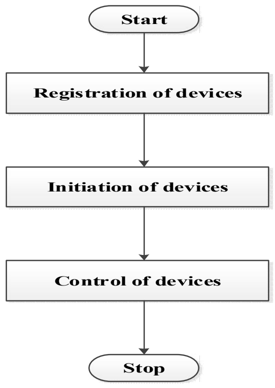
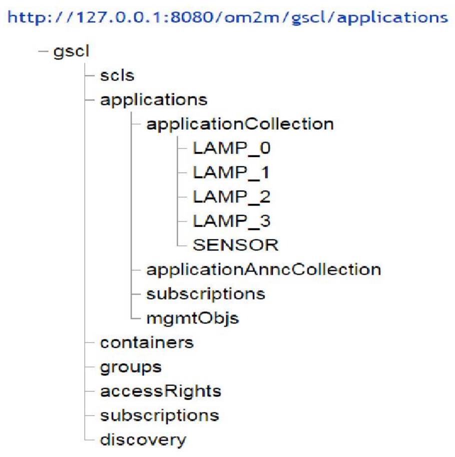


Figure 9. Procedure for controlling the electrical devices



```
http://127.0.0.1:8080/om2m/gscl/applications

  - gscl
      scls
      applications
          applicationCollection
              LAMP_0
              LAMP_1
              LAMP_2
              LAMP_3
              SENSOR
          applicationAnncCollection
          subscriptions
          mgmtObjs
      containers
      groups
      accessRights
      subscriptions
      discovery
```

Figure 10. Registered devices on app database at M2M platform

The figures 11 - 15 are some screen-shorts and pictures taken from the implemented system and application. Figure 11 is the sensor board designed and assembled by our team for temperature measurement while figure 12 illustrates measured temperature displayed on the GUI of the application. And finally the figure 13 shows the control board of the application that allows users to do the controlling activities like switch on or switch off the registered devices.

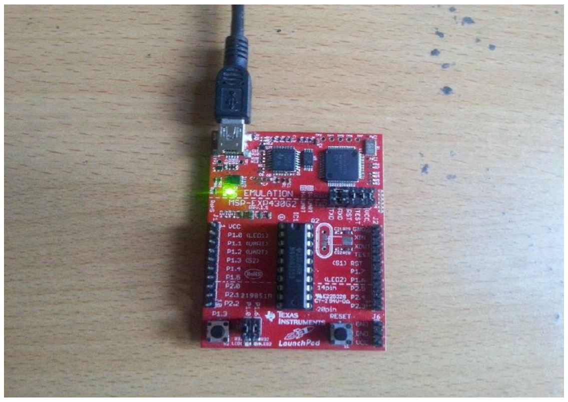


Figure 11: The sensors board

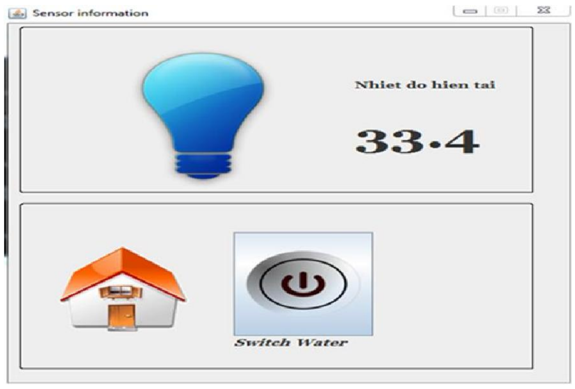


Figure 12: The information of measured temp.

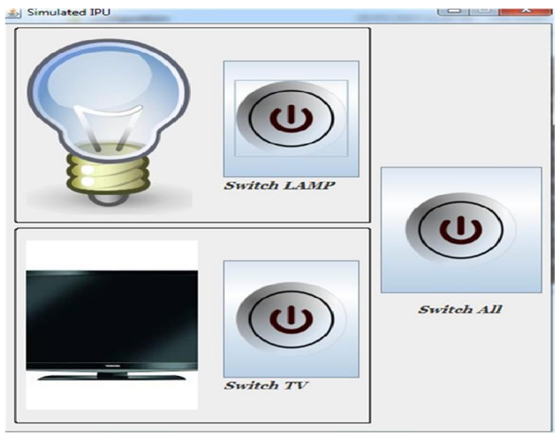


Figure 13: The control board for administrator/user

IV. 3. Performance Evaluation

With the test-bed depicted in Figure 3 we have tested two applications: the Home Automation (HA) and the Smart Meter. The HA application is described on section IV-B above while the Smart Meter is application that make use of the Arduino-board based data transceiver to collect electrical usage data (Figure 14) to send back to the platform and ultimately to back-end server.

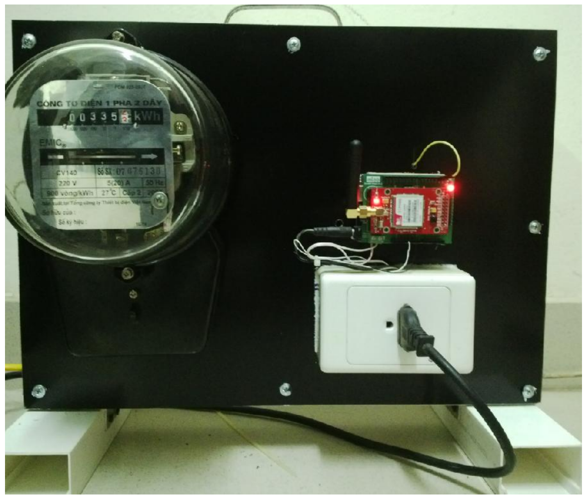


Figure 14: The Smart Meter Model for electrical usage data collection

The data collected here includes *electrical usage level, identifier and full name of the user, identifier of the meter, location information, and date and time interval of the data collection*. The test shows that the platform can communicate smoothly to semi-auto meter and with high reliability and low latency as depicted on Table 1.

Table 1. Performance evaluation of the platform for Smart Meter application

Item	**Time interval**	**Latency**	**Transmi-ssion cycle**	**No. Of transmis-sion**	**Loss Times**	**Loss Ratio**
1	10 days	1s	60’	240	5	2,08%
2	10 days	1,5s	60’	240	2	0,8%
3	10 days	2s	60’	240	1	0,4%

The test is conducted over 30 days divided into 3 intervals of 10 days each. The data is sent back every 60 minutes. The results show that the transmission latency and loss ratio are quite reasonable and if we want the small latency then the loss ratio may increase and vice versa.

We have also developed and tested few application communication protocols (those are analyzed on section III-B) on our platform and test-bed. The results show that CoAP is the most lightweight protocol (least number of messages exchanged per session and also smallest message size) so it is suitable for the sensors/actuators that need to save energy. But if the battery is not an issue then HTTP is better fit with RESTful applications due to its capability to carry rich information. The table 2 below summarizes the details of protocols implementation.

Table 2. Details of protocol implementation

Protocol	Transport	QoS	Procedure	No. of messages
CoAP	UDP	YES	Request/Response	03
RESTful	HTTP	NO	Request/Response	06
XMPP	TCP	NO	Publish/Subscribe	05
MQTT	TCP	YES	Publish/Response	05

V. CONCLUSION AND FUTURE WORK

This paper presents our comprehensive analysis of the M2M architecture as well as its wide range of applications in the real life. Our research has pointed out the necessity of completing the architecture and implementation of the various processes (core processing module, protocols, service exposing layer, device management and more importantly the API for application development) in order to allow the M2M technology to be deployed in real life environment. The proposed implementation for the M2M platform makes use of the advanced concept of service combination with usage of the standard architecture and protocols. These results pave firm initial steps on the way toward completing the whole IoT (Internet of Things) ecosystem that make M2M communication come in to real life applications such as smart home, smart city, fleet management, etc. The research is in fact our on-going work; we currently work on the more sophisticated mechanisms for device management as well as processing of the collected data from wide range of sensors/actuator and enhancing the system control mechanisms.

REFERENCES

[1] ETSI Machine-to-Machine Communications info and draft http://docbox.etsi.org/M2M/Open/ Retrieved: Dec, 2012

[2] Emmerson, Bob. "M2M: the Internet of 50 billion devices" *WinWin Magazine* (2010): 19-22.

[3] I. Cha, Y. Shah, and A. U. Schmidt, “Trust in M2M Communication”, IEEE Vehicular Tech. Mag., vol. 4, no. 3, Sep. 2009, pp. 69-75.

[4] M. Starsinic, “System Architecture Challenges in the Home M2M Network”, in Proc. Applications and Tech. Conf., Long Island, USA, May 2010.

[5] http://www.onem2m.org/

[6] R. Lu, X. Li, X. Liang, X. Shen, and X. Lin “GRS: The green, reliability, and security of emerging machine to machine communications”, Communications Magazine, IEEE, April 2011, Vol. 49 , No. 4, pp. 28-35

[7] D.Uckelmann, M.Harrison, and F. Michahelles “An Architectural Approach Towards the Future Internet of Things” ARCHITECTING THE INTERNET OF THINGS, 2011, pp. 1-24, DOI: 10.1007/978-3-642-19157-2_1

[8] A. de Saint-Exupery, “Internet of Things – Strategic Research Roadmap”, Sep.15, 2009. http://www.internet-of-things-research.eu

[9] J. C. Ferreira, R. Roque, C. Roadknight, J. Foley, P Ytterstad, and B. Thorstensen. Sensor Telco “new business opportunities; Deliverable 1 - Main technology trends, capabilities of devices and service examples” Technical report, Eurescom, 2006

[10] Angelo P. Castellani, Mattia Gheda, Nicola Bui, Michele Rossi, Michele Zorzi, “Web Services for the Internet of Things through CoAP and EXI”, IEEE International Conference on Communications Workshops (ICC), 5-9 June 2011, pp. 1-6.

[11] Maria Rita Palattella, Nicola Accettura, Xavier Vilajosana, Thomas Watteyne, Luigi Alfredo Grieco, Gennaro Boggia, Mischa Dohler, “Standardized Protocol Stack for the Internet of (Important) Things”, Communications Surveys & Tutorials IEEE 15(3), 2013, pp. 1389-1406.

[12] Shinho Lee, Hyeonwoo Kim, Dong-kweon Hong, Hongtaek Ju, “Correlation Analysis of MQTT Loss and Delay According to QoS Level”, International Conference on Information Networking (ICOIN), 28-30 Jan. 2013, pp. 714-717.

[13] Dinesh Thangavel, Xiaoping Ma, Alvin Valera, Hwee-Xian Tan, Colin Keng-Yan Tan, "Performance Evaluation of MQTT and CoAP via a Common Middleware", IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 21-24 April 2014, pp. 1-6.

[14] Sven Bendel, Thomas pringer, Daniel Schuster, Alexander Schill, Ralf Ackermann, Michael Ameling, "A Service Infrastructure for the Internet of Things based on XMPP", IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), 18-22 March 2013, pp. 385-388.

[15] Roy Thomas Fielding, "Architectural Styles and the Design of Network-based Software Architectures", PhD thesis, University of California, Irvine, USA, 2000.

[16] http://en.wikipedia.org/wiki/Advanced_Message_Queuing_Protocol, cited 28 Jul 2014.

[17] Joel L. Fernandes, Ivo C. Lopes, Joel J. P. C.Rodrigues, Sana Ullah, "Performance Evaluation of RESTful Web Services and AMQP Protocol", Fifth International Conference on Ubiquitous and Future Networks (ICUFN), 2-5 July 2013, pp. 810-815.

[18] http://en.wikipedia.org/wiki/WebSocket, cited 28 Jul 2014.

[19] R. Fielding, "Architectural styles and the design of network- based software architectures", Phd thesis, 2000.

[20] Cesare Pautasso, Olaf Zimmermann, and Frank Leymann, "Restful web services vs. big web services: making the right architectural decision", In Proc. of the 17th international conference on World Wide Web (WWW '08), pages 805–814, New York, NY, USA, 2008. ACM.

[21] Draft ETSI TS 102 690 V0.13.3 (2011-07) Technical Specification.

[22] J. Yim, Y. Choi, and B. Lee "Third Party Call Control in IMS using Parlay Web Service Gateway", Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference, 20-22 Feb. 2006, pp. 221 – 224.

[23] McAffer J, VanderLei P, Archer S., "*OSGi and Equinox: Creating Highly Modular Java Systems",* Addison-Wesley; Upper Saddle River; NJ; 2010.

[24] http://openmobilealliance.org/about-oma/work-program/device-management/

[25] http://www.etsi.org/plugtests/COAP2/Presentations/03_ETSI_M2M_oneM2M.pdf

AUTHOR BIOGRAPHY

Nguyen Tai Hung (PhD) is a lecturer of the Faculty of electronic & telecommunication at the Hanoi University of Science & Technology in Hanoi, Vietnam. He got his Master and PhD degrees, both in communication engineering, from same university in 2001 and 2007, respectively. He is the (co)author of around thirty of scientific papers/articles and a national research project in the fields of Internet Engineering and Next Generation Networks on international journals and conference proceeding. He spent a half of year in 2008 with Fraunhofer Institute of Fokus in Berlin, Germany for conducting the research project of service development for 3G/NGN networks.

Based on his more than 10 years of experience in the teaching complex IT and telecommunication technologies to different courses, from beginning to the most advanced levels, Dr. Nguyen Tai Hung gains an extensive knowledge in field of the Next Generation Networks and the Future Internet.