Dissertação de Mestrado

Inatel

Instituto Nacional de Telecomunicações

Performance Assessment of Management Protocols and Platforms for Internet of Things

JONATHAN DE CARVALHO SILVA

Dezembro, 2018

Performance Assessment of Management Protocols and Platforms for Internet of Things

JONATHAN DE CARVALHO SILVA

Dissertation presented to the National Institute of Telecommunications (INATEL), as part of the requirements to obtain the Master's Degree in Telecommunications.

Advisor:
Prof. Dr. Joel José Puga Coelho Rodrigues

Santa Rita do Sapucaí 2018

Silva, Jonathan de Carvalho
S586p
Performance Assessment of Management Protocols and Platforms for Internet of Things. / Jonathan de Carvalho Silva. – Santa Rita do Sapucaí, 2019.
101 p.

Orientador: Prof. Dr. Joel José Puga Coelho Rodrigues.
Dissertação de Mestrado em Telecomunicações – Instituto Nacional de Telecomunicações – INATEL.
Inclui bibliografia e anexo.

1. Internet of Things 2. Heterogeneity 3. IoT Network Management 4. IoT Device Management . 5. Platforms. 6. Protocols 7. Simple Network Management Protocol 8. SNMP 9. NETCONF 10. M4DN.IoT 11. Mestrado em Telecomunicações. I. Rodrigues, Joel José Puga Coelho. II. Instituto Nacional de Telecomunicações – INATEL. III. Título.

CDU 621.39

Ficha Catalográfica elaborada pelo Bibliotecário da Instituição
CRB6-2718 – Paulo Eduardo de Almeida

ATA Nº 174: DEFESA DA DISSERTAÇÃO DE MESTRADO EM TELECOMUNICAÇÕES – ÁREA DE CONCENTRAÇÃO: ENGENHARIA ELÉTRICA

Autor: Jonathan de Carvalho Silva

No dia 19 de dezembro de 2018, às 15h, na sala prof. José Nogueira Leite do Inatel, realizou-se a defesa de dissertação de mestrado em Telecomunicações, cuja área de concentração é Engenharia Elétrica, do Engenheiro Jonathan Carvalho Silva, intitulada "Performance Assessment of Management Protocols and Platforms for Internet of Things". A banca julgadora foi composta por: Prof. Dr. Joel José Puga Coelho Rodrigues do Inatel, presidente, Prof. Dr. Ricardo de Andrade Lira Rabelo da Universidade Federal do Piauí – UFPI e Prof. Dr. Guilherme Augusto Barucke Marcondes do Inatel. Estiveram presentes, além dos componentes da banca e do mestrando, professores, funcionários, alunos do Inatel e outras pessoas. O presidente deu início aos trabalhos, anunciando ser esta a centésima septuagésima segunda defesa de dissertação do Curso de Mestrado em Telecomunicações do Inatel. Solicitou ao mestrando proceder a sua defesa, o que foi feito no tempo regulamentar. Em seguida, os membros da banca examinadora fizeram perguntas, solicitaram esclarecimentos e teceram comentários sobre o trabalho desenvolvido. Terminada a fase de arguição e debates, os membros da banca reuniram-se em caráter reservado para a deliberação quanto ao resultado da defesa:

(X) Aprovada sem restrições, mas condicionada às eventuais revisões indicadas pelos membros da banca examinadora
( ) Aprovada com restrições e condicionada às revisões indicadas pelos membros da banca examinadora
( ) Reprovada

O presidente anunciou o resultado aos presentes e eu Gisele Moreira dos Santos, secretária do Curso de Mestrado em Telecomunicações, lavrei a presente ata que, aprovada, foi assinada pelos membros da banca examinadora. Santa Rita do Sapucaí, 19 de dezembro de 2018.

[illegible]
Prof. Dr. Joel José Puga Coelho Rodrigues

Ricardo de Andrade Lira Rabelo
Prof. Dr. Ricardo de Andrade Lira Rabelo

[illegible]
Prof. Dr. Guilherme Augusto Barucke Marcondes

"One machine can do the work of fifty ordinary men. No machine can do the work of one extraordinary man."

Elbert Green Hubbard, Be Thankful

Dedication

“*To God, to my parents João Cândido
Ribeiro da Silva Filho and Carmen
Lúcia de Carvalho Silva and Carolina
for their unconditional support at all times.*”

Acknowledgments

I thank God first for giving me health and intelligence to complete this work.

I would like to express my sincere thanks to my supervisor, Prof. Joel Rodrigues, who was at my side throughout the development of the work, for his confidence, optimism, and guidance through all stages and challenges of this work.

To Prof. Guilherme Marcondes and Prof. Carlos Nazareth Marins, thank you very much that during this master's work, they encouraged and made available resources to carry out this work.

My thanks to the group, IoT Research Group, for helping me and letting me contribute to the group and also to the friendships gained during this learning process.

Thank you for supports of Finatel through the Inatel Smart Campus project, and by Finep, with resources from Funttel, grant no. 01.14.0231.00, under the Radiocommunication Reference Center (Centro de Referência em Radiocomunicações — CRR) project of the National Institute of Telecommunications (Instituto Nacional de Telecomunicações — Inatel), Brazil.

I would like to say a big thank you to all my family and my girlfriend, who have always been present in my life, whether in times of difficulties and sorrows or in times of joy. I am sure they will always be together in my walk.

Finally, I thank to my parents, João Cândido Ribeiro da Silva Filho and Carmen Lúcia de Carvalho Silva who are the people who are most enthusiastic about my personal and professional success.

Contents

List of Figures

List of Tables

Abbreviations

3PP	Third Party Product
ASN.1	Abstract Syntax Notation One
BER	Basic Encoding Rules
BGP	Border Gateway Protocol
COMAN	COnstrained netwoks and devices MANagement
CoAP	Constrained Application Protocol
CSV	Comma-Separated Values
DaaS	Database as a Service
DFD	Data flow diagram
DSD	Data Structure diagrams
DTLS	Datagram Transport Layer Security
FCAPS	Fault, Configuration, Accounting, Performance, Security
ERD	Relationship Entity Diagrams
GUI	Graphical User Interface
IDE	Integrated Development Environment
IETF	Internet Engineering Task Force
IoT	Internet of Things
IoT-A	IoT Architecture
IoT-PIC	Internet of Things Platform’s Infrastructure for Configurations
IoT RA	Internet of Things Reference Architecture
ISCampus	Inatel Smart Campus
ISO	International Organization for Standardization
ITU-T	International Telecommunications Union - Telecommunications
JMX	Java Management Extension
JSON	JavaScript Object Notation
JWT	JSON Web Tokens
LNMP	LoWPAN Network Management Protocol
M2M	Machine-to-Machine
ManIoT	Management for the Internet of Things
MANET	Mobile Ad Hoc Networks
MD-SAL	Model Driven Service Abstraction Layer
MIB	Management Information Base
MO	Mobile Object
MQTT	Message Queuing Telemetry Transport
NED	Network Enabled Devices
NETCONF	Network Configuration Protocol

NFV	Network Functions Virtualization
NoSQL	Not Only SQL
OID	Object Identifiers
OMA-DM	Open Mobile Alliance Device Management
OMA-LwM2M	Open Mobile Alliance Lightweight M2M
OpenNMS	Open Network Management System
OVSDB	Open vSwitch Database
PAN	Personal Area Network
PCEP	Path Computation Element Protocol
QoS	Quality of Service
RA	Architecture Reference
RRD	Round Robin Database
SDN	Software-Defined Network
SNMP	Simple Network Management Protocol
SOA	Service-Oriented Architecture
TMP	Things Management Protocol
UDP	User Datagram Protocol
UML	Unified Modeling Language
WSN	Wireless Sensor Network
WSP	Wireless Session Protocol
XML	eXtensible Markup Language
XMPP	Extensible Messaging and Presence Protocol

Resumo

A Internet das Coisas (do Inglês, Internet of Things - IoT) é um paradigma no qual os dispositivos estão conectados à Internet e podem interagir uns com os outros, realizando um cenário de comunicação máquina a máquina. Os ambientes de IoT precisam de escalabilidade, comunicação padronizada e requisitos de reconhecimento de contexto para obter o gerenciamento de dispositivos conectados com segurança e precisão em ambientes reais. O gerenciamento eficiente de redes IoT exige considerar as restrições dos respectivos dispositivos de baixa potência e a complexidade de implantação da infraestrutura de comunicação subjacente. Um cenário de gerenciamento IoT é representado por um número crescente de dispositivos conectados, caracterizado por sua heterogeneidade, além dos protocolos e plataformas de gerenciamento de rede e dispositivos IoT. Um dos desafios das redes IoT está relacionado à heterogeneidade dos dispositivos, em que cada objeto possui diferentes capacidades de processamento, memória e tecnologias de comunicação. No entanto, as redes IoT geralmente são caracterizadas por dispositivos com recursos restritos e, às vezes, implantadas em áreas de difícil acesso. Além disso, uma rede IoT pode incluir um grande número de dispositivos heterogêneos que necessitam ser gerenciados. Todos esses fatores determinam a necessidade de soluções de gerenciamento adaptadas para dispositivos e redes IoT. Para garantir um melhor uso das redes, é necessário um gerenciamento eficaz e automático. Esta dissertação estuda diferentes soluções para gerenciamento de rede e dispositivos num contexto IoT, identificando suas características e desafios técnicos. Com relação aos protocolos de gerenciamento de rede e dispositivos IoT, a maioria das soluções tenta resolver o problema de economia de energia e simplicidade devido aos recursos limitados de hardware dos dispositivos. Houve o desenvolvimento de outras soluções adaptadas para um ambiente IoT, porém os protocolos *Simple Network Management Protocol* (SNMP) e *Network Configuration Protocol* (NETCONF) continuam a ser usados em grande escala por equipamentos comerciais. Dentre as plataformas de gerenciamento IoT, a plataforma OpenDayLight obteve vantagem em relação às outras plataformas por ser uma plataforma modular capaz de gerenciar tanto a rede quanto os dispositivos e por dar suporte aos protocolos nativos, nomeadamente, SNMP, NETCONF, *Constrained Application Protocol* (CoAP) e OpenFlow. No entanto, atualmente, na literatura, não há plataformas de gerenciamento que abordem todas as questões relacionadas com IoT. Essas tecnologias foram estudadas e suas características discutidas em detalhe. Com base no estudo comparativo das soluções existentes foi proposta e criada uma nova plataforma de gerenciamento IoT com uma interface amigável, chamado M4DN.IoT (*Management for Device and Network in Internet of Things*). Essa plataforma modular integra e gerencia as funcionalidades individuais dos dispositivos em uma rede IoT, bem como o estado e as características dessa rede. Além disso, a plataforma fornece serviços genéricos, como descoberta de nós, armazenamento de dados e autenticação, que são requisitos básicos para a criação de aplicativos IoT. A solução foi avaliada, demonstrada, validada e encontra-se pronta para utilização integrada na plataforma de *middleware* para IoT, a In.IoT.

Palavras-chave: Internet das coisas; Internet of Things; Heterogeneidade; Gerenciamento de Rede IoT; Gerenciamento de Dispositivos IoT; Plataformas; Protocolos; Simple Network Management Protocol; SNMP; NETCONF; M4DN.IoT.

Abstract

Internet of Things (IoT) is a paradigm where devices are connected to the Internet and can interact with each other, making a scenario of machine to machine communication. IoT environments need scalability, standardized communication, and context-aware requirements for managing securely and accurately connected devices in real-world environments. Efficient management of IoT networks requires consideration to the constraints of their low-power devices and the complexity of deploying the underlying communication infrastructure. An IoT management scenario is represented by a growing number of connected devices, characterized by their heterogeneity more network management protocols and platforms, and IoT devices. One of the IoT network challenges is related to devices heterogeneity where each object has different processing capabilities, memory capacity, and communication technologies. However, IoT device networks are generally characterized by devices with restricted resources and sometimes deployed in hard areas to reach. In addition, an IoT network may include a large number of heterogeneous devices that need to be managed. All these factors determine the need for management solutions tailored for devices and the IoT network. To ensure better use of IoT networks, effective and automatic management is required. This dissertation studies different solutions for network management and devices in an IoT context, identifying their characteristics and technical challenges. With respect to network management protocols and IoT devices, most of the solutions try to solve the problem of energy saving and simplicity due to the basic hardware features of the devices. There are been development of other solutions tailored for an IoT environment, but Simple Network Management Protocol (SNMP) and Network Configuration Protocol (NETCONF) continue being used on a large scale by commercial equipment. Among the IoT management platforms, OpenDayLight has gained advantage over other platforms because it is modular and capable for managing both the network and devices to support native protocols, namely, SNMP, NETCONF, Constrained Application Protocol (CoAP), and OpenFlow. However, currently, in the literature there are no management platforms that address all IoT-related issues. These technologies have been studied and discussed in detail. Based on the comparative study of the available solutions, a new IoT management platform with a friendly user interface, called M4DN.IoT (Management for Device and Network in Internet of Things), was proposed and created. This modular platform integrates and manages the individual functionalities of the devices in an IoT network as well as the state and characteristics of that network. In addition, the platform provides generic services, such as node discovery, data storage, and authentication, which are basic requirements for creating IoT applications. The solution has been evaluated, demonstrated, validated, and is ready for use in the middleware platform for IoT, the In.IoT.

Keywords: Internet of Things; Heterogeneity; IoT Network Management; IoT Device Management; Platforms; Protocols; Simple Network Management Protocol; SNMP; NETCONF; M4DN.IoT.

Chapter 1

Introduction

1.1 Motivation

Initially, computer networks were created for communicating as a mean of sharing endpoint devices with the same standards of networks, protocols, and operating systems. However, the fast evolution of networks combined with a reduction of computational resources costs motivated the increase of computer networks in all the markets [1]. Considering this scenario, it becomes increasingly necessary to manage the network environment to keep it working properly. Network management is required to maintain the entire network structure working, thus meeting the user needs and the administrators' expectations.

Due to the emergence of Internet of Things (IoT), it is expected an exponential growth of network endpoint devices (NEDs) becoming a challenge in the areas of infrastructure, security, energy saving, among others [2]. The continued growth in the number and diversity of network components has also contributed to the fact that network management activity has become more and more indispensable. The benefits of integrating a company's computing systems of different nature and sizes as a way of distributing tasks and sharing available resources are now a reality. For this reason, an efficient data management system is required by IoT networks so the information is always available wherever and whenever requested [3].

IoT management presents two scopes: the devices and the network. In each of them, there is a huge variety of protocols and management platforms to minimize the challenges (presented in this work). Given the number of existing protocols and platforms, an evaluation should determine which IoT management protocols are capable of efficiently satisfying the application requirements and which platforms support these protocols for real environments (real deployment).

Communication between devices performing machine-to-machine communication (M2M), Wireless Sensor Networks (WSNs) for monitoring and control processes, and the interconnection of WSNs with the Internet are examples of some challenges for managing an IoT network [4]. Network devices using software-based communication (known as software defined networks - SDN) gathers, detects, and configures data from sensors, thus creating the context of managing a network. New technological approaches focusing on IoT are emerging, as Fog/Cloud technologies [5]. They are compatible with constrained portable devices and with old management protocols, therefore, being an IoT trending topic. Figure

1.1 presents a typical scenario involving different communication technologies and a gateway where connected devices collect information from the environment (e.g., temperature, luminosity, movement, and others) and report data to an IoT management network entity.

One of the most important challenges in this IoT scenario is the network device heterogeneity [6] [7]. Devices can support different communication protocols with different formats and data types, memory and processing capacity [8]. Another important factor is the data set produced in real time and the implicit semantics imposing challenges regarding the configuration and infrastructure of IoT environments [9]. An illustration of the heterogeneity of medium access control (MAC) protocols for IoT is shown in Figure 1.1.

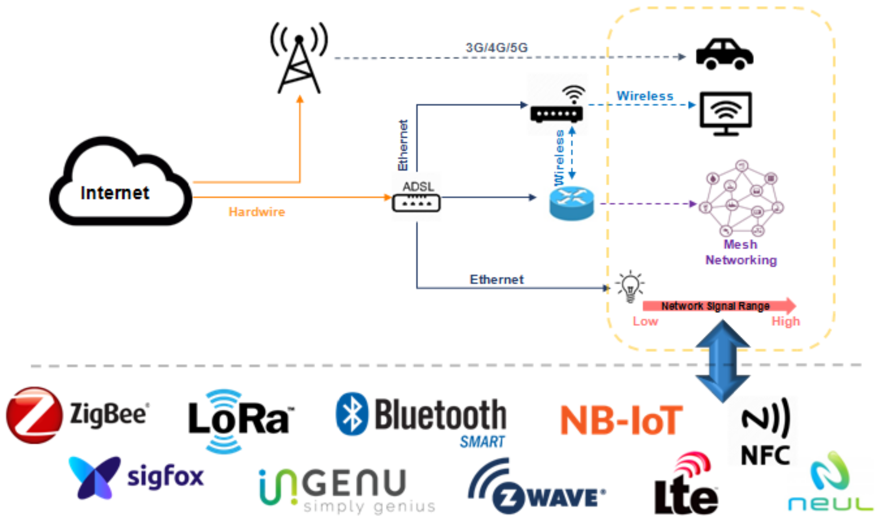


FIGURE 1.1: Illustration of an IoT Network Architecture and a plethora of available protocols.

The complexity of IoT network management compared to traditional TCP/IP networks management is also greater than WSNs [10]. IoT needs to support networking devices and services that involve *i)* the use of a plethora of devices with diverse characteristics, and *ii)* the IoT networks devices interaction through local or remote management context aware. WSNs should manage frequent communications failures and low security of wireless links (i.e, the MANNA architecture [11]), and this management must also be context aware. The available IoT management architectures partially attend these features [12]. Then, this dissertation focuses on available policies, approaches, and solutions, including tools for IoT management. Among the available technologies for management, it was described and performs a comparison study considering their heterogeneity, scalability, supported technologies, security, among others. Based on this evaluation, the most promising technologies were chosen for a detailed performance evaluation study (through simulation and deployment in real environments) and the relevant features are used to create a new IoT management platform, called M4DN.IoT (Management for Device and Network in an Internet of Things).

1.2 Problem Definition

IoT Management systems need scalability, standardized communication, and context-aware requirements to achieve the management of connected devices with security, and accuracy in real environments [13]. Interoperability and heterogeneity between hardware and application layers are also critical issues [14].

Typically, IoT Network Management has complex middleware/platforms that make use of protocols without a standard to support communication and the integration of the connected device on IoT network management. Thus, the evaluation and comparison of such platforms and protocols to choose the best technologies is a hard task since they are hardly comparable as a consequence of their lack of standardization [15].

Considering the above-mentioned issues, this dissertation focuses on IoT-based platforms and protocols for network and device management aiming to select the best candidate among platform(s) and protocol(s) for use in real environments. In other words: ***given the current scenario of heterogeneity in networks and devices, it must be possible to propose a new IoT Management solution that should gathers contributions from the best approaches available in the current management platforms and protocols***.

1.3 Research Objectives

The main objective of this dissertation includes the performance evaluation of IoT network management protocols, devices management solutions, and platforms to propose a solution and corresponding validation in real environments. Thus, it could identify possible aspects to improve in order to propose and create a new IoT management solution. To attain this main objective, the following partial objectives were identified:

- Deep review of the state of the art on IoT management protocols and platforms;
- Performance evaluation of available IoT network management protocols and identification of possible aspects to improve;
- Proposal and development of a new solution for IoT management to be deployed in a real environment;
- Performance evaluation, demonstration, and validation of the new solution in a real environment.

1.4 Main Contributions

The first contribution of this dissertation is the in-depth state of the art review of the Internet of things management platforms and protocols. This review thoroughly analyzed the architectures and features of the protocols and platforms as well as their associated challenges and problems. This study is described in detail in Chapter 2. A preliminary review was published in the Simpósio *Brasileiro de Telecomunicações e Processamento de Sinais*

(SBrT 2017) [16] and a survey paper submitted for publication in an international journal [17].

The second contribution is focused on the performance evaluation of IoT management protocols in an IoT real environment with a scenario of a smart lighting monitoring and management. This performance evaluation study uses qualitative and quantitative metrics in order to determine the best network protocols for an IoT scenario. This proposal solved the problem of interoperability and heterogeneity of devices and network management. This contribution is described, in detail, in Chapter 3, and it was published in a paper presented in the IEEE International Workshop on Communication, Computing, and Networking in Cyber Physical Systems (CCNCPS 2018), IEEE International Conference on Communications (IEEE ICC 2018) [18].

The third contribution includes the development, performance evaluation of IoT management platforms, using an IoT real environment, with a scenario of a smart lighting monitoring and management. The OpenDayLight platform has a modular design and supports IoT devices management and IoT network management. The OpenNMS is monitoring and management platform that only provide support of the IoT network management. The platforms (OpenDayLight and OpenNMS) were evaluated according to the packet size, latency, throughput, and average response time metrics. The experiments and results analysis concluded that OpenDayLight platform is the most suitable and robust for real environments. Its results were better than the other platforms and hosts both SNMP and NETCONF protocols natively. This contribution is included in Chapter 3, and was included in a paper presented at the International Conference on Advances in Computing, Communications and Informatics (ICACCI 2018) [19].

Finally, the last contribution of this dissertation proposes the modeling, development, evaluation, demonstration, and validation of a new IoT management solution, called M4DN.IoT. It is a modular platform that manages the state and quality of the connections between network devices that determine an IoT network management solution. This proposal solved the problem of standardizing the data format used in communication between applications, services, and devices. This contribution is detailed described in Chapter 4 and it is included in a paper submitted for publication in an international journal [20] [21].

1.5 Publications

During this research work, several research papers were prepared. They are listed below.

- **Jonathan de C. Silva**, Joel J. P. C. Rodrigues, Kashif Saleem, Sergei A. Kozlov, Ricardo A. L. Rabelo, “M4DN.IoT - A Networks and Devices Management Platform for IoT”, in IEEE Access Journal, ISSN: 2169-3536 (in press). ISI Journal Citation Report with impact factor of 3,557 in 2017.

- **Jonathan de C. Silva**, Joel J. P. C. Rodrigues, Jalal Al-Muhtadi, Ricardo A. L. Rabêlo, Vasco Furtado, "Management Platforms and Protocols for Internet of Things: A Survey". Sensors 2019, Volume 19, Issue 3, pp. 676, February 1, 2019, https://doi.org/10.3390/s190306

- **Jonathan de C. Silva**, Pedro H. M. Pereira, Lucas L. de Souza, Carlos N. M. Marins, Guilherme A. B. Marcondes, Joel J. P. C. Rodrigues, “Performance Evaluation of

IoT Network Management Platforms", 7th International Conference on Advances in Computing, Communications and Informatics (ICACCI 2018), Bangalore, India, September 19-22, 2018.

- Lucas L. de Souza , Pedro H. M. Pereira, **Jonathan de C. Silva**, Carlos N. M. Marins, Guilherme A. B. Marcondes, Joel J. P. C. Rodrigues, "IoT 5G-UDN Protocols: Practical Model and Evaluation", IEEE International Workshop on Communication, Computing, and Networking in Cyber Physical Systems (CCNCPS 2018), in association with IEEE International Conference on Communications (IEEE ICC 2018), Kansas City, MO, USA, May 20-24, 2018.
- **Jonathan de C. Silva**, Joel J. P. C. Rodrigues, Mario Lemes Porença Jr, "IoT Network Management: Content and Analysis", Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT 2017), São Pedro, Brazil, September 3-6, 2017.
- **Jonathan de C. Silva**, Joel J. P. C. Rodrigues, Antonio M. Alberti, Petar Solic, Andre L. L. Aquino, "LoRaWAN - A Low Power WAN Protocol for Internet of Things: a Review and Opportunities", 2nd International Multidisciplinary Conference On Computer and Energy Science (SpliTech 2017), Split, Croatia, July 12-14, 2017.
- Thiago Santos da Costa, **Jonathan de C. Silva**, Joel J. P. C. Rodrigues, Ricardo A. L. Rabêlo, Neeraj Kumar, Petar Solic. "Performance Assessment of Software Defined Networks Management Protocol in Real Environments". Submitted to an international conference.
- **Jonathan de C. Silva**, Flávio B. Brito, Joel J. P. C. Rodrigues, Guilherme A. B. Marcondes. "Estudos sobre Plataformas de Gerenciamento de Redes e Dispositivos para Internet das Coisas". Submitted to XXXI Congresso de Iniciação Científica do Inatel.

1.6 Registered Software

The created IoT management platform, called M4DN.IoT, was registered and its reference is the following (in Portuguese):

- **Jonathan de C. Silva** and Joel J. P. C. Rodrigues, "M4DN.IoT – Management for IoT Devices and Networks", Pedido de Registro de Programa de Computador - RPC N^{o} BR 512018051860-5, Outubro, 2018.

1.7 Thesis statement

The choice of an IoT management solution is based in a survey focusing on a deep review of the promising platforms and protocols for IoT management aiming the improvement, performance evaluation, and deployment on real environments to obtain the device and network controls when the interoperability and the technological diversity of the connected devices are critical requirements. This relevant features and improvements are considered to create a new IoT management solution, called M4DN.IoT.

1.8 Document Organization

This remaining chapters of this document are organized as follows. Chapter 2 surveys the state of the art focusing the on characteristics and analysis of IoT network and device management platforms and protocols. Open research issues on the topic are identified.

Chapter 3 describes the real environment used for the performance evaluation of the IoT management platforms and protocols to study the features to propose and develop in a new IoT management platform.

Chapter 4 includes the architecture and requirements description of the new IoT management platform (M4DN.IoT). In addition, the evaluation, demonstration, and validation of the created platform is presented.

Chapter 5 concludes the dissertation, showing the lessons learned, final considerations, and suggestions for further studies.

Chapter 2

IoT Management Protocols and Platforms

In this chapter, the state of the art on IoT management was elaborated. It presents a deep study of the related literature on network and device management, and the latest developments on the topic.

2.1 Background

Computer networks are composed by heterogeneous communication devices and sharing resources [22]. Computer networks management emerged after a rapid evolution of network technologies, in addition to a major effort to reduce the costs of computing resources [23]. The offered services range from simple resources sharing to current technology and assume that every object can be connected to the Internet. This is known to as the Internet of Things (IoT). Network management goals controlling and monitoring network elements, physical or logical, ensuring a certain quality of service (QoS) level. To accomplish this task, define network management as a collection of tools for monitoring or managing devices [24]. The traditional network management model can be summarized as follows: *i)* data collection from monitoring managed resources automatically, *ii)* diagnosis to analyze and solve identified problems throughout the monitored data, and *iii)* action or control to solve a problem or modify the state of a device [25].

For Kurose *et al.* [25], a network without management mechanisms can present problems such as interference in data traffic, lack of data integrity, high congestion rates, resources that can be misused or overloaded, as well as security problems. According to Gabdurahmanov [26], network management can be difficult for three reasons: *i)* the managed network is heterogeneous because it contains hardware and software components manufactured by various companies; *ii)* technology can change continuously with new services available; and *iii)* the managed networks are large and the network nodes may be distant from other nodes.

Kurose *et al.* [25] states that ISO has created an IoT reference architecture (RA) in which network management includes five functional areas, as follows: performance management, failure, configuration, accounting, and security (as shown in Figure 2.1). Performance management intends to analyze, measure, inform, and control the performance of different network components, i.e., routers and hosts. In failure, the purpose is to log, detect, and react to network failure conditions. The division between fault management and performance management is undefined [27]. Failure management can occur through transient network failures, such as the interruption of service on a link or routing software. Performance management, however, takes a long term approach.

The administrator can know the managed network devices and their hardware configurations through management actions [28]. Accounting is intended to allow the network administrator to specify, register, and control access to users and devices of the network. Quotas and usage charges for privileged access to the network resources make up the accounting management. According to organization policies defined, the network resources access is performed with security management.

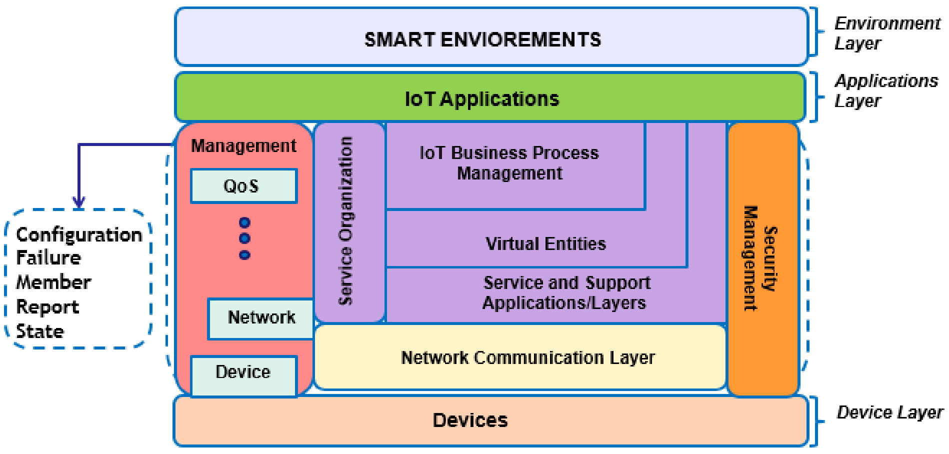


FIGURE 2.1: Internet of Things Reference Architecture (IoT-RA).

In Figure 2.2, the general architecture of network management systems presents the following four basic components: elements, stations, protocols, and information on network management [25]. They are briefly described below.

Agent: The managed elements have a software that allows monitoring and controlling the equipment through one or more management stations;

Manager: Management station communicates with the agents, either for monitoring or controlling them. Usually, the management station offers an interface through which authorized users can manage the network;

Protocol: The standard protocol, used for operations of monitoring (reading) and control (writing); it is necessary for information exchange between manager and agent devices;

Management information: The management information has the data that can be referenced for operations by the management protocol, i.e., the managers and agents can exchange data to obtain information such as the SNMP protocol [29].

Since the proposal of IoT years ago, many ideas have had three main constraints that restrict its development [30]: *i)* proprietary communication protocols, *ii)* security and privacy, and *iii)* inconvenience to manage. Thus, the work focuses on network and device management and if it is feasible to address management constraints.

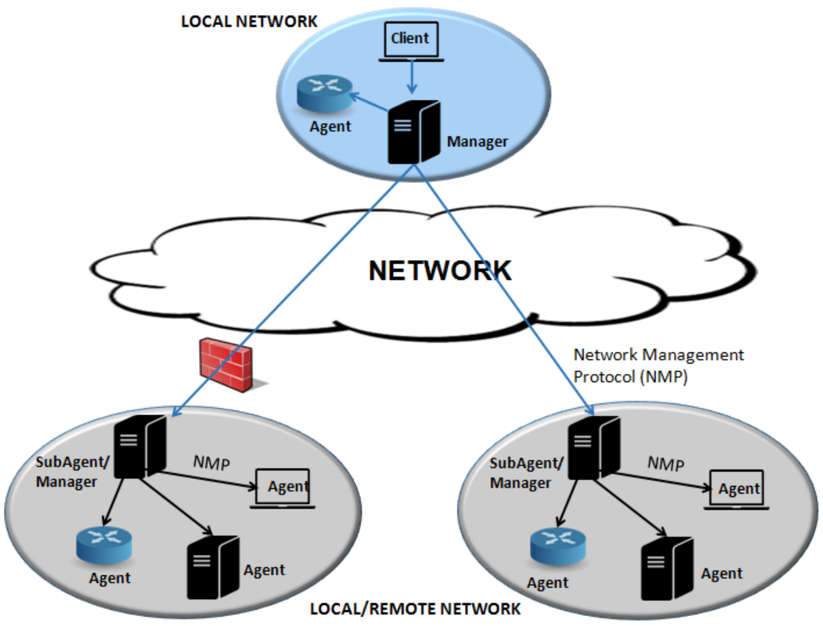


FIGURE 2.2: Components of network management systems.

Due to the specific characteristics and challenges of IoT, devices cannot be managed using only the traditional management tools. Thus, IoT management has two categories: IoT Network Management and IoT Device Management. IoT Network Management is required to collection and analysis large volumes of data from IoT platforms and, consequently, provides efficient decisions and/or actions. IoT Device Management is required to provide the device location and status information, e.g., update embedded software, disconnect some stolen or unrecognized device, modify security and hardware configurations, locate a lost device, and even enable interaction between devices.

IoT Network Management often needs to adapt the unknown topology of these networks by providing device location and status information. Managing services should have the capacity to disconnect and locate lost devices, modify security settings, delete device data, and more. Delicato *et al.* [31] states that management should consider the possibility of integrating and using the devices not previously planned in the environment opportunistically.

An IoT network environment can have various connected devices in the same network, such as a health sensor, a control/medical server, a Web report of statistics and a smartphone, as illustrated in Figure 2.3. Thus, it is important that a management platform enables the devices to dynamically detect other devices present in the environment to meet the requirements of the applications.

The main difference between WSNs, AdHoc networks (MANETs), and IoT networks is the characteristic of heterogeneous devices and topologies performed by these networks [32]. All the time, new devices compose these networks, therefore, the IoT management platforms require a customized module (driver) to translate device functionality into the platform. Some platforms for WSN and AdHoc networks can be used in IoT networks.

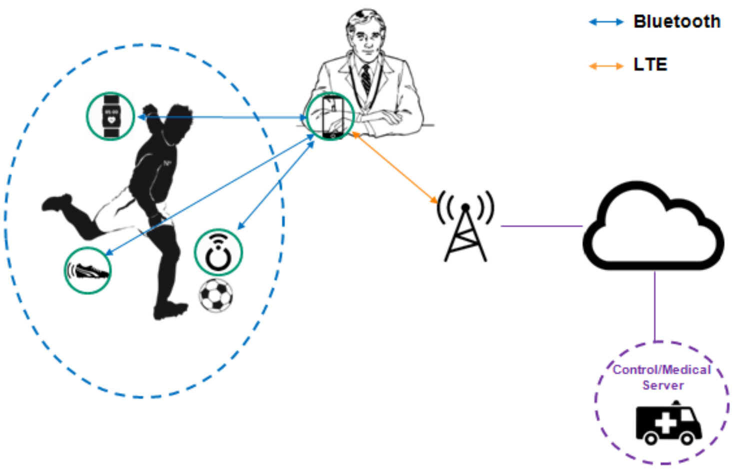


FIGURE 2.3: Illustration of an IoT network environment.

IoT management solutions must meet some requirements [12]. For example, there must be interoperability between platforms and network devices. The platform must obtain the connected devices dynamically through discovery and management. A solution should be context-aware and support scalability (considering indications of intensive usage). Security and dynamic adaptation should keep data integrity and privacy guaranteeing devices availability and QoS.

Existing IoT management platforms, however, only partially meet these requirements [12]. Due to lack of standardization, IoT management needs to specify data and information models, in which these models are used to define a format for storing and deploying other management services. In addition to the above requirements and other aspects such as security, authentication and authorization, there are characteristics in the scope of local and global management which are discussed in [33].

In the IoT reference architecture [34], network management includes functional components. Configuration (self-configuration) is responsible for initializing system configuration, i.e, collect and store the device configurations, tracking configuration changes, and planning future extensions.

The self-aware component, Failure, identifies, isolates, recovers, and records failures in an IoT system. For each occurrence of a failure, a notification is sent to the *Failure component* with the objective of collecting more data in order to identify the type and degree of the problem.

Member is responsible for monitoring and recovering members. This component allows recovering members of the system while obeying a certain filter and allows the subscription to receive updates of register/unregister member meta-data in the database.

Report refines and maintains the history of the information provided by management devices, e.g., to determine the efficiency of the current system through the collection and analysis of performance data.

State goals self-monitoring of the IoT system with the past, present, and future devices states. It is required by the Failure component, having the functions of changing or applying a particular state in the system. It also checks the consistency of commands provided for this function and monitors the state, which makes it possible to predict and update the state for a certain time or to recover the state of the system through a history.

2.2 IoT Network Management Protocols

The section elaborates on the most relevant IoT network management protocols considering the Simple Network Management Protocol (SNMP), Network Configuration Protocol (NETCONF), Open vSwitch Database (OVSDB), Internet of Things Platform's Infrastructure for Configurations (IoT- PIC), IEEE 1905, and LoWPAN Network Management Protocol (LNMP).

2.2.1 Simple Network Management Protocol (SNMP)

The objective of the SNMP is to find and fix the bugs or problems of a network [35]. Through SNMP agents, the network administrator can view network traffic statistics and is able to change its configurations after analyzing this data. Defined at the application level and standardized by the IETF [36], the SNMP uses the User Datagram Protocol (UDP) transport protocol to send messages over the network without delivery guarantee.

Over the last years, other protocols have used the same concept, e.g., NETCONF was created to replace SNMP. The SNMP continues to dominate the network management market, mainly because of its simplicity of implementation, since it consumes fewer network resources and processing, which allows the inclusion of very simple equipment. According to SNMP Research International *et al.* [37], there was an incremental development of three versions of the SNMP, as described below:

- SNMPv1: it offers a management solution with low cost and simple implementation, but with a lack of authentication and security mechanisms and limitation in the performance of messages in very large networks.

- SNMPv2c: it created the management of decentralized networks, allowing the existence of more than one management station and, consequently, the exchange of information between them. In SNMPv2, it was not possible to reach a consensus regarding the security standard to be used in SNMP, and there was an addition of other request types such as *getrequest, get-next-request, set-request, response,* and *snmpV2-trap*.

- SNMPv3: it published a set of documents defining a framework to incorporate security features into full capacity (with the SNMPv1 or SNMPv2 features). The SNMP architectural model includes a management server and network devices. Servers monitoring and control the network devices with events. The network devices include equipment as hosts, gateways, terminal servers, that have managing agents used to be handled by a server. The SNMP protocol is used to exchange data between a server and network devices.

Figure 2.4, shows some of the possible interactions between manager and agent through the SNMP protocol [38]. An SNMP device can be connected to other devices, performing machine-to-machine communication [39].

The SNMP Agent is a software installed in a device to support network management. It answers the queries from SNMP managers and sends a trap message to some events

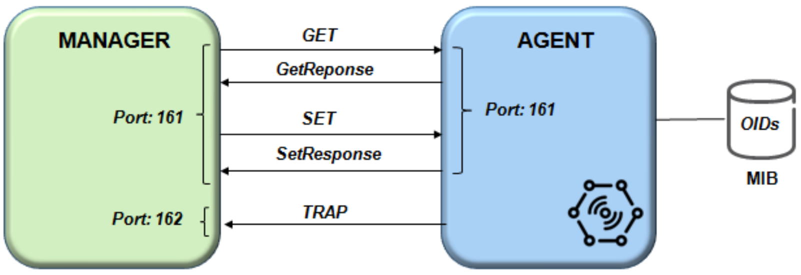


FIGURE 2.4: Illustration of the SNMP Protocol Architecture.

(according to their priority). The Management Information Base (MIB) is a virtual database with object identifiers (OIDs) organized in a tree structure to keep information about device management in a communication network.

The ASN.1 (Abstract Syntax Notation One) notation [40] is a language developed by ITU-T and chosen by ISO for the definition of the MIB manageable objects. It uses object-oriented concepts to define a resource, so that its attributes can be performed by this resource, when applicable.

The SNMP is a non-connection-oriented protocol does not require a prior or subsequent action to send messages. So the protocol messages will no guarantee the destination is reached. It is a simple and robust protocol, yet powerful enough to solve the difficult problems presented by managing heterogeneous networks such as an Internet of Things network, as shown in Figure 2.5. Therefore, the key problems to manage the sensors in IoT involves the MIB design and the development of manager and agent software.

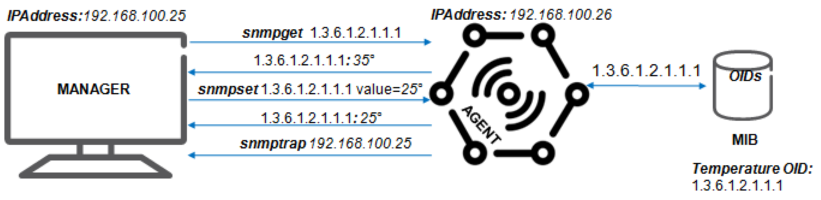


FIGURE 2.5: Illustration of the SNMP protocol for IoT devices.

Use Case

The SNMP protocol is a base unit that provides a centralized platform for operations and facility management teams to monitor sensor conditions and configure threshold-based alerts. It aims to monitor and collect data from environmental sensors and to integrate data from equipment such as generators and other devices enabled to integrate with intelligent sensors through SNMP, as demonstrated by OpenNMS, which is an open source platform [41].

2.2.2 Network Configuration Protocol (NETCONF)

The NETCONF is a protocol for network configuration and monitoring, as defined by the IETF [42] and, therefore, has better features than the SNMP, which had as its weakest point the absence of network configuration resources, i.e., the interface BER (Basic Encoding Rules) and proprietary MIBs. The NETCONF protocol was developed to be the natural successor to SNMP, because SNMP is focused on monitoring and not on network configuration while NETCONF uses mechanisms that allow the installation, manipulation, and removal of network device configuration through a client-server implementation, as shown in Figure 2.6.

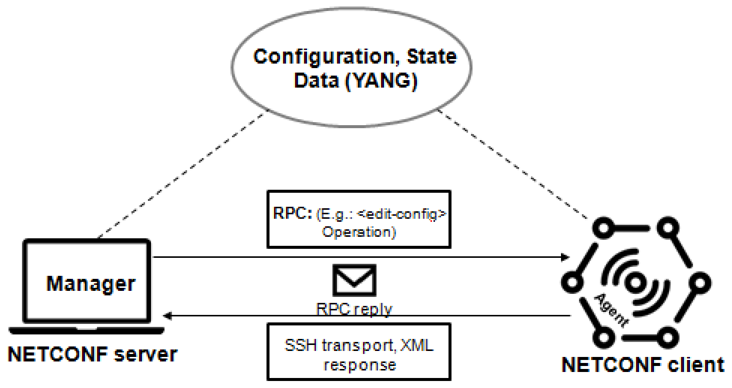

Configuration, State Data (YANG)
RPC: (E.g.: <edit-config> Operation)
Manager
RPC reply
NETCONF server
SSH transport, XML response
Agent
NETCONF client

FIGURE 2.6: Illustration of the NETCONF Protocol for IoT Architecture.

After establishing the secure transport session between client and server, the NETCONF protocol sends a *HELLO* message to announce the protocol capabilities and supported data models. NETCONF also supports the subscription and receipt of event notifications asynchronously as well as the partial closing of a current configuration of a network device. This feature allows multiple editing sessions, streamlining the configuration process. NETCONF allows monitoring and management of an autonomous entity (the NETCONF manager) that uses the repository of data, sessions, closings, and statistics available on the NETCONF server.

The NETCONF protocol transports this information to an application manager, who can infer the required settings for the network devices. YANG is a formal language with clear text of the data model with syntax and semantics that allow the construction of network applications [43]. The YANG model can be translated into an XML (Extensible Markup Language) or JSON (JavaScript Object Notation) file, structured in a tree for each module, with properties that correspond to the functionalities of the device and declarations of types, data, constraints, and additions of reusable structures.

Heterogeneous networks are characteristic of the Internet of Things, and the NETCONF protocol is used to efficiently manage and resolve issues of this network. Most operating systems developed for IoT [44] such as TinyOS and Contiki OS already have the NETCONF protocol built into their operating system.

Use Case

Currently, most vendors, e.g. Cisco, already use the NETCONF protocol on their equipment as a standard model. Another example, the OpenFlow devices controller communicates with connected devices in an SDN architecture, also defining a protocol for such communication. The OpenFlow provides means to control network devices using NETCONF, without the need for manufacturers to expose the code of their legacy products [45].

2.2.3 Open vSwitch Database (OVSDB)

Open vSwitch Database (OVSDB) is a management protocol in a software-defined network (SDN) environment [46]. Most network devices allow remote configuration using legacy protocols, such as SNMP. The goal of OVS consists in creating a modern programmatic management protocol interface – OVSDB.

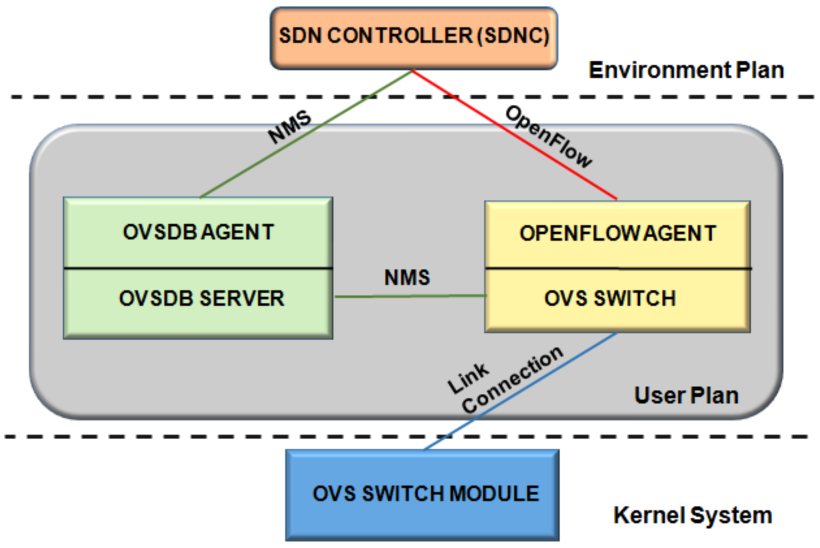


FIGURE 2.7: Illustration OVSDB Protocol for IoT Architecture.

According to Figure 2.7, the OVSDB management protocol handles Open vSwitch (OVS) that consists of a database server (OVSDB Server), a virtualized switch (OVS Switch) and, optionally, a module for fast-path forwarding. Each OVS is managed by, at least, one manager. An OVS module supports several data paths referred to as “bridges”, where this controller uses OpenFlow.

The OVSDB protocol executes the configuration and management operations on the OVS instance. OVSDB is used to create/delete/modify bridges, ports, and interfaces. The OVS represents an evolution of network management protocols, allowing programming and configuring bridges, ports, and interfaces for SDN equipment platforms and Network Functions Virtualization (NFV) [47].

Use Case

The open source OpenDaylight platform for Software Defined Networking (SDN) uses open network management protocols, i.e. SNMP, OVSDB, and NETCONF to provide modular functions, extensible control, and network device monitoring [48].

2.2.4 Internet of Things Platform's Infrastructure for Configurations (IoT-PIC)

The IoT-PIC allows network management to perform the platform commissioning installed in the network. The IoT-PIC architecture is similar to the SNMP protocol described in section 2.2.1, as shown in Figure 2.8. It has the possibility of performing any configuration or composition of hardware and software resources.

The IoT-PIC architecture have two levels, the global and local, and it is composed by two components, in which the communication among these components is made through the XMPP protocol [49]: an IoT-PIC Manager (PIC-M) at a global level and an IoT-PIC Agent (PIC-A) at a local level.

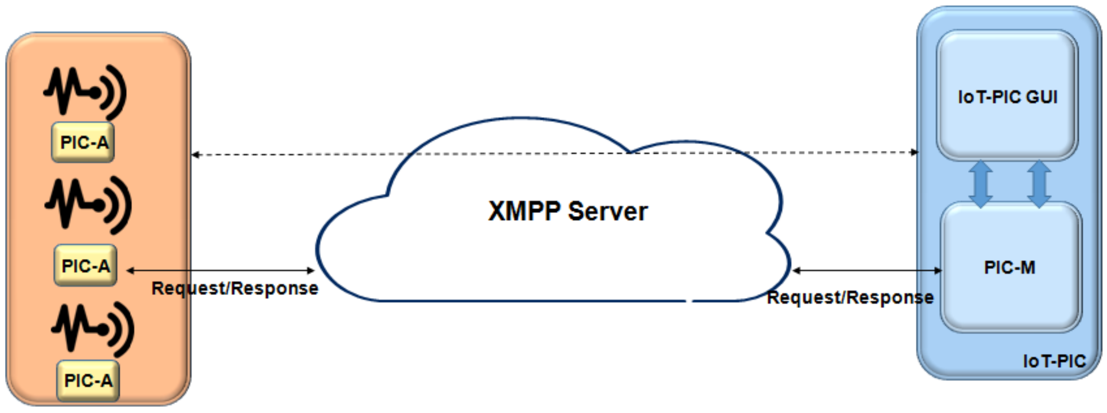


FIGURE 2.8: Illustration of the IoT-PIC architecture.

The XMPP protocol is an open-source IETF standard protocol based on XML for network management in IoT contexts, which allows real-time messaging, the information exchange and request/response services. The XMPP performance of latency, scalability, and robustness has been widely demonstrated during the years [50].

The PIC-M module is used to manage the configuration and access of the other modules to the platform. It interacts as an interface to applications and other platform components.

The PIC-M functionalities consist in notifying the applications on the status of the device available in the middleware, requesting configuration information from the PIC-A via "get" and "set", through an XMPP command, and updating the configuration of devices through PIC-A via XMPP.

The IoT-PIC uses the publish-subscribe, in which subscribers only receive messages of interest, without information on the publishers, which allows the complete decoupling of the devices. Each platform is associated with a PIC-A and responds to the management of the PIC-M device. The configuration and interconnection of devices are assigned to the PIC-A, e.g., adding and removing the connection.

The IoT-PIC deploys the discovery functionality of devices through the XMPP protocol. New devices connected are automatically registered to the network, describing their functionalities with a common format [51]. Particularly, in the proposed solution, when a new device connects to the network, the manager of this network publish-subscribe joins in PIC-M for all discovered resources.

A resource example is a sensor that measures humidity and temperature. Context Manager can create location-related nodes where devices can enter their location allowing navigation of the tree from the root. First, the PIC-M creates a collection node with a given device id, containing two nodes, in which the first node has the temperature function and the second node has the humidity function. The resources of the nodes used in the service discovered by PIC-M are associated to the resource types list, i.e. *Humidity Sensor* and *Temperature Sensor*. The functions of the nodes are associated to the operation list, i.e. *getTemperature* and *getHumididity*. This list creates the entire hierarchy of nodes and, if the user does not indicate the parameter, the entire list is returned.

Use Case

The IoT-PIC is used in an energy efficiency scenario. The IMPReSS platform [52] includes energy saving and alarm system applications to allow sensor, lights, and smart plugs into the platform. In order to save energy, the Energy Saver manages the light through the PIC-M and tells the PIC-A of the lights management component to publish/subscribe node of the device in order to receive its events; e.g., in a classroom, detects motion sensor if a row of seats is empty, in which case the lights are automatically switched off.

The GUI interface converts the XML returned by response of the PIC-M into a user-friendly form. This platform allows integrating new devices without need modifications to the deployment environment.

2.2.5 IEEE 1905

IoT environments depend on several MAC protocols. The challenge of interoperability between technologies needs to be discussed. IEEE 1905 is a standard focused on the convergence of digital home network and offers an abstraction layer to all these heterogeneous MAC protocols.

The goal of IEEE 1905 is to define a common standard that establishes home network technologies for a data and control service access point. Each interface can transmit and receive packets, regardless of underlying technologies or layers, as shown in Figure 2.9 [53].

An intermediate layer used to exchange messages (Table 2.1), is called Control Message Data Units (CMDUs), with all standards compatible devices. In Figure 2.10, all the IEEE 1905 deployed devices with Abstraction Layer Management Entity (ALME) protocol have neighbor discovery, topology exchange and rules, measured traffic, and security associations following the layers presented.

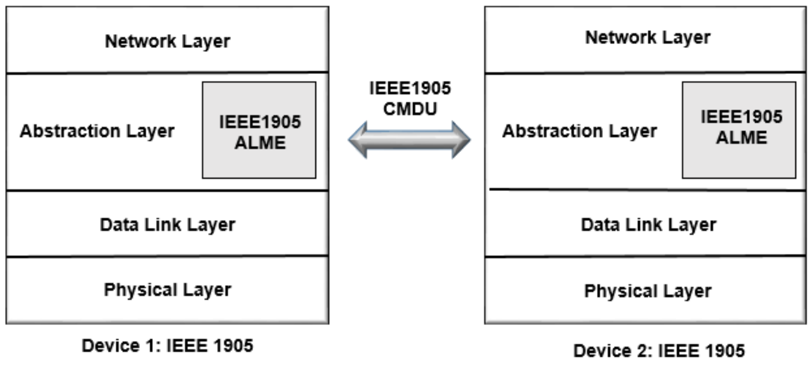


FIGURE 2.9: Illustration of the protocol structure for IEEE 1905.

TABLE 2.1: Exchanged messages at the Abstraction layer.

Exchange messages	**Description**
ALME-GET	This message is used by the HLE's to get a description of the HLE's device.
ALME-SET	This message is used by the HLE's to send a configuration of the HLE's device.

The protocol introduces an intermediate abstraction layer to the logical link control (LLC) and one or multiple media access control (MAC). The service access points (SAPs) holds many networking technologies, in order to support advanced network management features like auto-configuration, quality of service (QoS), path selection and discovery. This layer simplifies setup, e.g., by eliminating the need for a user to enter different passwords to access each link [54].

This ALME SAP entity is capable of providing management services to MAC, physical layer (PHY) and higher layer entities (HLEs) [55]. It also provides advanced network management features including discovery and interface selection.

Use Case

nVoy is an application of the IEEE 1905 standard program [56] that provides the services to maximize and simplify the overall performance of a home network. The reliability is provided through the abstraction layer to established power line, wireless, coaxial cable and Ethernet home networking technologies - IEEE 1901 / HomePlug® AV, Wi-Fi, MoCA®, and Ethernet, allowing to provide common setup procedures for establishing connected devices, secure links, and network management.

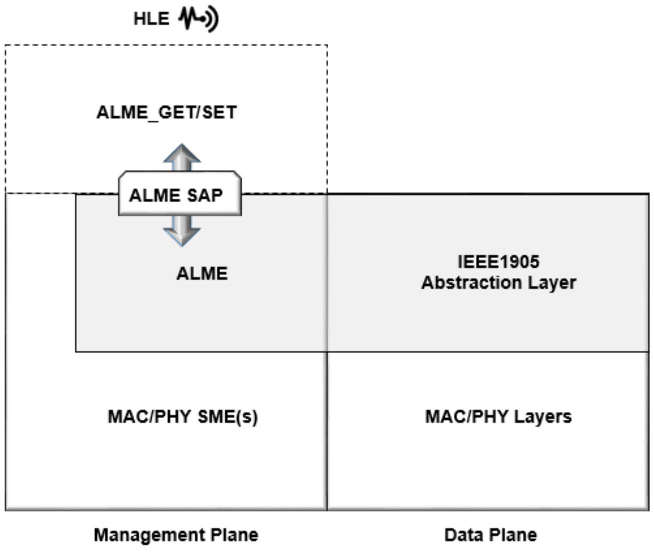


FIGURE 2.10: IEEE 1905 standard network management architecture.

2.2.6 LoWPAN Network Management Protocol (LNMP)

The LNMP is a management architecture suitable for 6LoWPAN networks [57]. With LNMP architecture is possible to reduce the cost communication and, therefore, increasing the lifespan of the network. The LNMP main characteristic is to allow interoperability with SNMP. In terms of communication and complexity, the SNMP is considered impracticable due to the limited device's resources.

This architecture (Figure 2.11) allows the discovery of devices in a network with help of the coordinators in the monitoring and management. The SNMP is an application layer adapted protocol to run over IPv6, so uses this protocol to the adaptation layer 6LoWPAN[58]. The popular solution NET-SNMP [59] includes the adapted IPv4 and IPv6 for IoT network.

Exists two successive management operations that entities within the 6LoWPAN performed. First, Network Discovery is executed to monitor the device state in the architecture. The second step, after discovered devices, is the management of available devices.

To discover "live" devices manually, intense use of the resources is needed, and thus, the Network Discovery is a procedure created for an automated network state discovery, which is necessary given the WSN characteristics for their continued deployment. In this proposal [60], the network discovery uses the automated monitoring of the network state distributed by a 6LoWPAN network. The coordinators responsible to maintain the information about device state and reporting of subordinate devices has the device discovery feature. Bandwidth is a scarce resource in a sensor network and this feature reduces communication costs. The sensing and processing are usually lower than the communication cost.

It is desirable to monitor the device's status within the 6LoWPAN in a standard management protocol, e.g., SNMP protocols. However, the bandwidth available is a factor limited for application layer payload [61]. Therefore, SNMP is inviable due complexity of transport and communication into 6LoWPAN networks. Nonetheless, the reuse of network protocols is a goal of the 6LoWPAN, especially because of the interoperability of devices with SNMP. The SNMP message is translated to a UDP-based query when arrives from an NMS. It contains identifiers objects that are retrieved by the device agent. Likewise, these objects are translated to SNMP format when arriving at the gateway.

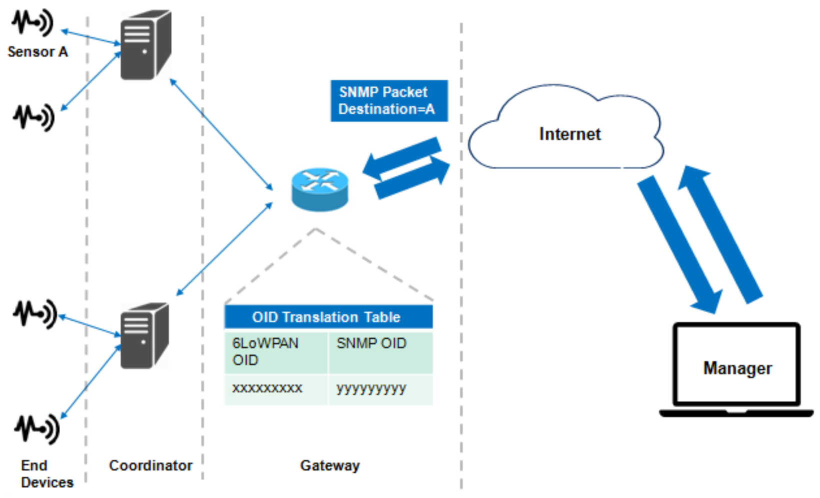


FIGURE 2.11: Illustration device level monitoring procedure for LNMP Protocol.

The data validity is the most important consideration to management architecture. The performance of the network management can be calculated with query-response delay and the increasing number of nodes. Likewise, another way is analysis the computation overhead with query load. The reliability introduces a delay of 25ms to a query and reaches up 50%. Queries with five hops proposed a delay of 100ms or more gave 100% reliability.

Use Case

In this proposal [60], the Internet Lab Ajou University deployed an agent application over the 6LoWPAN and a PAN coordinator connected to the gateway with PPP interface. The 6LoWPAN environment composed of a gateway and IEEE802.15.4 devices, containg a PAN coordinator. The devices support Hilow [62] routing protocol. The device management agent access to 802.15.4 information base, 6LoWPAN MIB, and IP MIB reduced.

2.3 IoT Network Management Platforms

This section describes the most relevant IoT network management platforms. IMPReSS, OpenNMS, OpenDaylight, and Zabbix were considered on this study.

2.3.1 IMPReSS

The IMPReSS project is a partnership between the European Union and Brazil (EU-Brazil).

The goal of the project was to provide a development platform that allows low-cost development of IoT complex systems and facilitates interaction with users and external systems [52]. The IMPReSS project ended on March 31, 2016.

The IMPReSS development platform can be used by any system that adopts the Smart Society context. The demonstration and validation of the IMPReSS platform will be carried out on energy efficiency systems to reduce the use of energy and CO_2 emission in public buildings. One contribution will be the inclusion of intelligence in monitor and control systems, as well as the stimulation of user awareness in reducing energy expenditures. For the configuration management, the PIC-A exposes two ad-hoc commands, in Table 2.2. The first command provides a list of management data, in XML format, associated with every variable, i.e., the type, the current value, and a list of values to assign. The second command updates values associated with a variable when this is writable [63].

TABLE 2.2: Configuration Management Ad-Hoc commands.

Ad-Hoc commands	**Description**
getAvailableDrivers	Returns the list of drivers available on the repository.
getConfiguration	This operation is used to get the list of the current values in the parameters that can be configured in the component.
setConfiguration	Updates the component configuration and setting the values passed as a parameter.

The application interacts with PIC-M that provides setConfiguration and getConfiguration commands to write and read the configuration in any PIC-A. When setting information in parameters, the setConfiguration should be called, passing the XML used to insert new configuration values.

2.3.2 OpenNMS

OpenNMS (Network Management System) open source platform [41] is used to the management and monitoring of business networks. Developed under the FCAPS (Fault, Configuration, Accounting, Performance, Security) network management model, it is distributed under the GPL license.

OpenNMS is written in Java, in addition to using database PostgreSQL or RRDTool, specifically JRobin (Java port for RRDTool), and supports Red Hat, Debian, Fedora, Mandriva, SuSE, Solaris, Mac OS X and Microsoft Windows.

The architecture presented in Figure 2.12 has the features to determine the availability and latency of services, storage and collecting of data, event management (such as SNMP traps), alarms and notifications.

It uses two flows for data collection in Round Robin Database (RRD). The first is through so-called monitors that connect to a network resource and perform a test to verify if it responds correctly. If this does not happen an event is generated. The second flow is

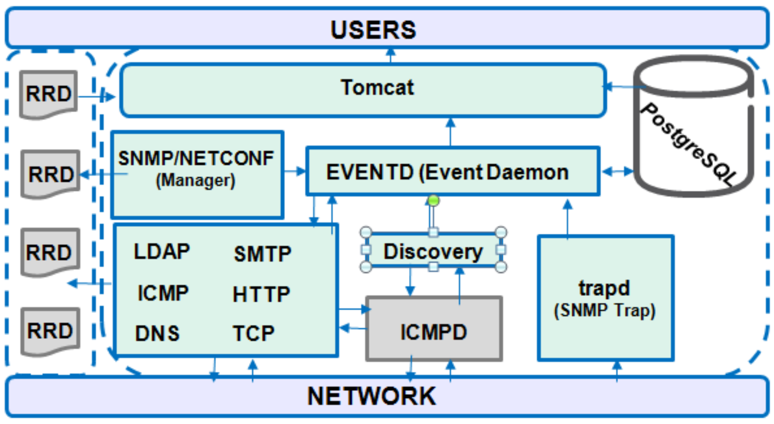


FIGURE 2.12: Illustration of the OpenNMS architecture.

through the use of so-called collectors, which can be collected by SNMP, NETCONF, Java Management Extensions (JMX), and HTTP.

The generated events are of two types; those generated internally by OpenNMS and those generated externally by SNMP traps, which are characterized according to their description and gravity [64].

2.3.3 OpenDaylight

The OpenDaylight (ODL) is an open source Web-platform for network management as Software Defined Networking (SDN). It uses open protocols to allow centralized control and network device monitoring [48]. The ODL supports OpenFlow and also offers ready-to-install modular network solutions. There is support for a wide range of network protocols, including SNMP, NETCONF, RESTCONF, OVSDB, BGP, PCEP, LISP, and more. OpenDaylight is slightly different from other controllers because it offers other protocols such as southbound interfaces, e.g., OpenFlow, BGP, and PCEP. In addition, OpenDaylight offers interfaces with OpenStack and Open vSwitch (OVSDB).

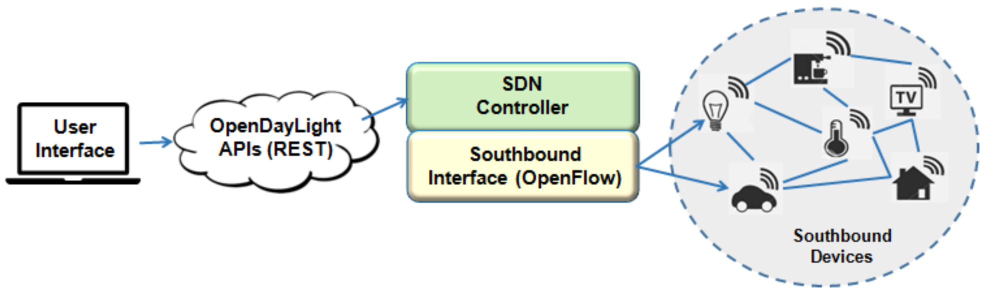


FIGURE 2.13: Illustration of the OpenDaylight architecture.

OpenDaylight is a micro-service that uses the sharing of YANG-based (NETCONF) data structures for messages exchange and data storage, as shown in Figure 2.13. According to Haleplidis *et al.* [65], through a model addressed to the Model Driven Service Abstraction

Layer (MD-SAL), can aggregate any application or function to a service and loaded by the controller.

2.3.4 Zabbix

Zabbix is an open source tool distributed under the GNU GPLv2 license for network management. It monitors the network services status as well as servers or other hardware. As described in [66], it is characterized as being a centralized management system with semi-distributed monitoring. In Figure 2.14, the organization can be divided into three main modules.

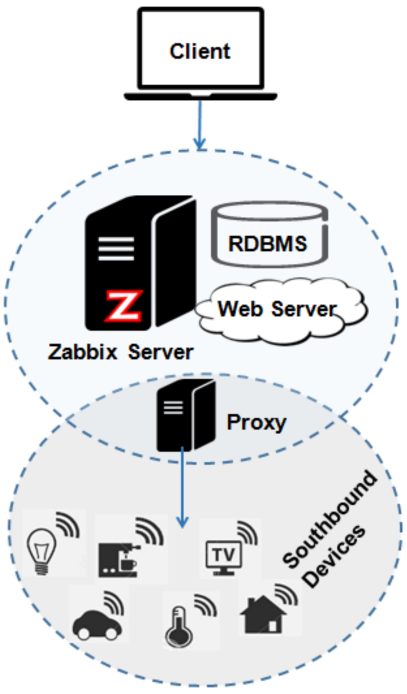


FIGURE 2.14: Illustration of the Zabbix architecture.

The platform architecture is distributed and consists of a central server in charge of administering the system and dealing with the interaction between the other two main components: *i)* the “Zabbix Agent” to monitor local resources and applications and send them to the server, and *ii)* the "Zabbix Proxy" is an optional part of the Zabbix configuration essential for distributed monitoring [67]. Zabbix proxy collects the data from the hosts and stores them in a database of their own in order to avoid loss of information if there is a problem with the communication with the server. The alert system includes three channels for sending notifications via e-mail, SMS and jabber (currently called XMPP - Extensible Messaging and Presence Protocol).

2.4 IoT Device Management Protocols

Device management has two main components: *i)* Device Manager and *ii)* Device Agent. The Device Manager is a system that communicates with devices through multiple management protocols and provides individual and bulk device controls. It also manages the device to block remotely when necessary [68].

According to Zehao Liu *et al.* [69], the Device Agent is a generic component suite that provides management of devices and utilities such as: *i)* communication adapters for HTTP and MQTT; *ii)* registration of devices; *iii)* token management, and *iv)* type of management platform.

The managed devices need to maintain and map the device's identity to their owners. Thus, it allows management through installed software, enabling/disabling functions, monitoring the device availability, and control the security features. Other functions should show be the location and, if available, locking the device remotely, among others. Unmanaged devices haven't any management agent and can communicate with the network. Semi-managed devices implement some parts of the managed devices, e.g., only feature control, but not software management [70].

2.4.1 COnstrained networks and devices MANagement (COMAN)

The COMAN Group from IETF [71] proposes Mobile Object (MO) solutions that simplified MIB, SNMP-based on messages, and CoAP-based management, which could be the protocol used for management of constrained networks and devices.

In Table 2.3, some device management candidate technologies were identified and described:

TABLE 2.3: COMAN - Candidate Technologies.

Technology	**Description**
CoAP	IETF has defined a binary protocol, the Constrained Application Protocol (CoAP), easy to analyze and specially designed for constrained devices, which is used with lower-level protocols, but it is particularly adapted over UDP/IPv6.
OMA-LwM2M	OMA Lightweight M2M is a device management protocol used to M2M networks environment.
OMA-DM	OMA Device Management provides functions for device management. The device management happens through communication between a server (Device Manager) and the client (Device Agent) using HTTP transport.

This survey limits the study to CoAP, OMA-LwM2M, and OMA-DM, but there are several candidates for COMAN technologies.

CoAP - COnstrained networks and devices MAnagement

CoAP is an easy to use protocol intended for devices with constrained resources and in conformation with the Rest Style. It is a specialized web transfer protocol designed for M2M applications. It was developed to be used along with lower level protocols and has been used in many IoT candidates along with IPv6 and UDP.

Also, this protocol meets most requirements for COMAN, such as group-based provisioning, capability discovery, support for energy optimized protocols, unreachable devices and lossy links [71] [72].

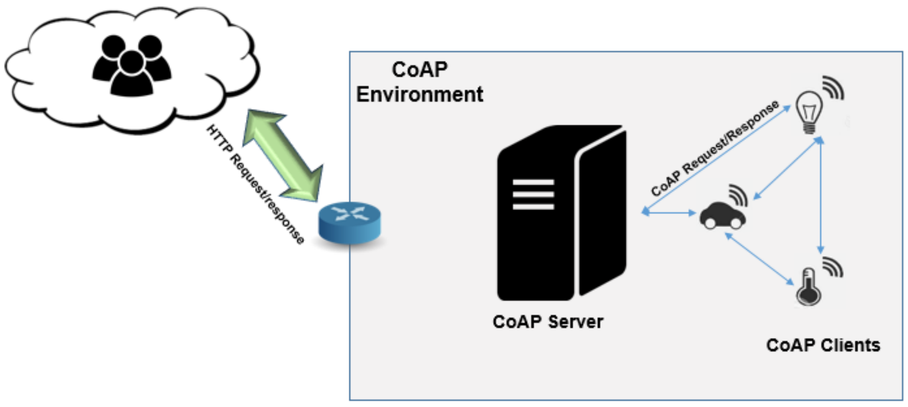


FIGURE 2.15: Illustration of the CoAP architecture.

As shown in Figure 2.15, the CoAP architecture abstracts all network elements as resources, called Universal Resource Identifier (URI) [73]. Inside CoAP management features, it can detect, with low complexity, if a device is online with a simple CoAP ping and verify if the server is stateless.

Also, in the fog computing architecture [5], it is possible to see the performance of this protocol compared to NETCONF and SNMP. This protocol can be used along DTLS (Datagram Transport Layer Security) [74].

OMA-DM - Open Mobile Alliance Device Management

OMA-DM provides the management information for connecting devices with the DM tree model [75], [76], [77] and remotely managing connected devices through the OMA-DM management protocol [78]. It provides efficient methods to manage connected "things" in network environments using: *i)* configuration maintenance and management, *ii)* configuration of user preferences, *iii)* fault detection, query and reporting, *iv)* non-application software download, *v)* provisioning, and *vi)* software management.

The OMA Device Management is divided into DM Server and DM Client devices [79]. The standard format for communication messages and data transports uses the XML format for the following technologies: physical layers lines or wireless networks (GSM, IrDA, Ethernet or Bluetooth) and transport layers over Wireless Session Protocol (WSP)/WAP [80], HTTP [81], OBEX [82] or similar transports, as shown in Figure 2.16.

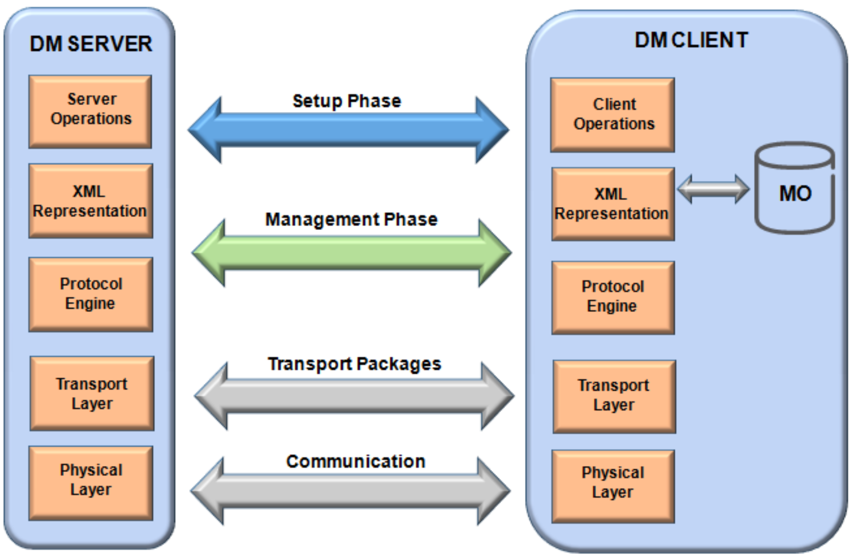


FIGURE 2.16: OMA DM standard management architecture.

OMA DM performs data exchange and device management with XML data through a DM server/client communication [83]. The OMA DM consists of two phases: *i)* a configuration phase, after authentication enables the exchange of device information through the user commands (Add, Alert, Copy, Get, and others) sent to the DM Client; *ii)* the management performs the request/response messages (Status, Generic Alert, and Results) between DM server/client.

OMA-LwM2M - Open Mobile Alliance for Lightweight M2M

The OMA LWM2M enables M2M device management, acting as an OMA-DM successor using the same protocol, and provides a compact and secure communication for this management [84]. It provides a sub-layer to allow management of LWM2M Server/Client, using a CoAP client-server architecture over UDP as a transport layer, as shown in Figure 2.17.

The M2M Service Provider, Network Service Provider, or Application Service Provider can be hosted by the LWM2M Server that provides a private or public data center [85]. The LWM2M Client is integrated into a software or device [86]. The LWM2M communication model [87] uses the CoAP methods (GET, PUT, POST, and DELETE) with bindings over UDP transport layer.

Use Case

Nowadays, there are several solutions (CoAP, OMA-LwM2M, or OMA-DM) with COMAN requirements, e.g., energy states, logging, system authentication, peripheral management, and access controls to the system [71]. Sprint is globally one of the best examples of a mobile Operator that has made FOTA part of its services strategy.

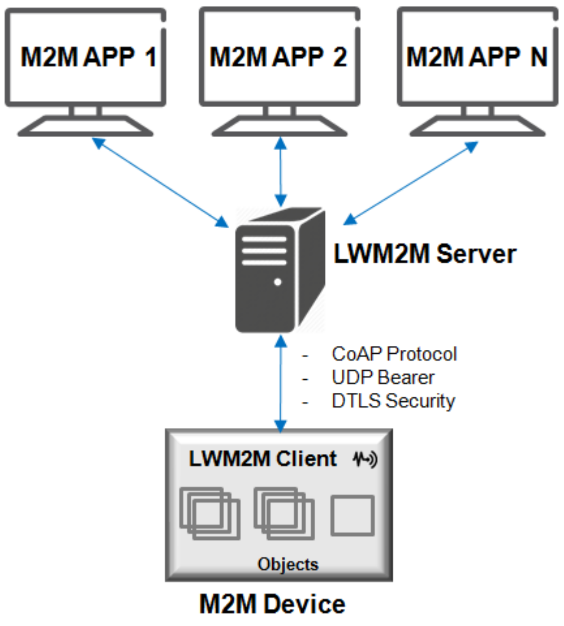


FIGURE 2.17: OMA LWM2M standard management architecture.

It is fully committed to providing FOTA updates according to the OMA DM standards [88].

2.4.2 Things Management Protocol (TMP)

The Things Management Protocol (TMP) uses the operations get/set, similar to the Simple Network Management Protocol (SNMP) operations, to enable default interface for communication between the *"things with things"* and *"things with the applications"* [89].

Guiping *et al.* [90] describes that the motivation for creating TMP was the need to manage the heterogeneity devices independently. TMP is SOAP-based, as shown in Figure 19, and uses key technologies such as HTTP, XML, and SOA for information integration and connection application based on independent protocols.

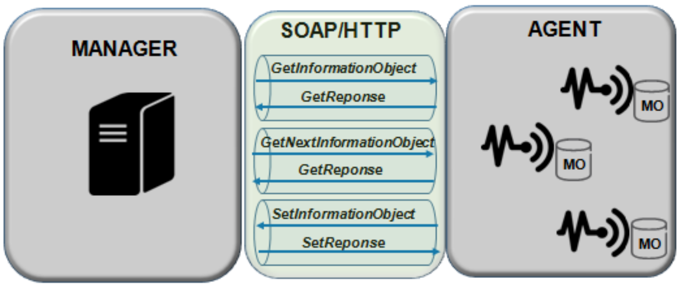


FIGURE 2.18: Illustration of the Things Management Protocol architecture.

In Table 2.4, the TMP creates the connection between protocol and transport layer protocols and includes several operation request/response messages in the protocol, e.g., GetInformationObject and SetInformationObject.

TABLE 2.4: TMP Operational Requests/Response Messages.

Operational Messages	**Functions**
GetInformationObject	In IoT application, this message is used to read things information object.
GetNextInformationObject	Used to read one or more things information object next to the current object.
SetInformationObject	Used to write one or more things information object. The value of one thing information object is written per one operation.

TMP supports three operations: GetInformationObject, GetNextInformationObject, and SetInformationObject. The basic requirements for operating "things information" are satisfied in these operations.

Use Case

The Smart Street Lighting System [91] can be managed remotely using Thing Management Protocol and some tasks can be automated with the objective of reduction of the power consumption, which has an ecological implication.

2.4.3 CPE WAN Management Protocol (CWMP)

Technical Report 069 (TR-069) is a specification that defines an application layer protocol for device management. It was initially published by Broadband Home Working Group and received the name of CPE WAN Management Protocol (CWMP). CWMP is an IP-based protocol and uses XML for all messages, as presented in Figure 2.19. It provides transaction confidentiality over Transport Control Protocol (TCP) with Secure Sockets Layer (SSL) or Transport Layer Security (TLS) and allows levels of authentication. The protocol uses Hypertext Transfer Protocol (HTTP) and Simple Object Access Protocol (SOAP) based on web services. The data models standardized and security methods are advantages of CWMP over SNMP.

This protocol works between CPE (Customer-premises equipment) and the Auto Configuration Server (ACS), achieving better scalability and cost reduction results. Many CPEs can be managed simultaneously by ACS because the session starts and short times are reserved for CPE [92]. The security of this protocol depends on ACS [93] [94]. A problem with this protocol is the scalability of the high volume of CPE for a single ACS. Thus, there is a proposal of addiction the components of the ACS management architecture using dynamic grouping, and sub-ACS structure [95].

The ACS can control the CPE through the *get* and *set* methods as parameter values. In the first message, the CPE sends CPE information, e.g., identification, manufacturer, and serial number to the ACS. Then, ACS sends a request with parameters for CPE to execute.

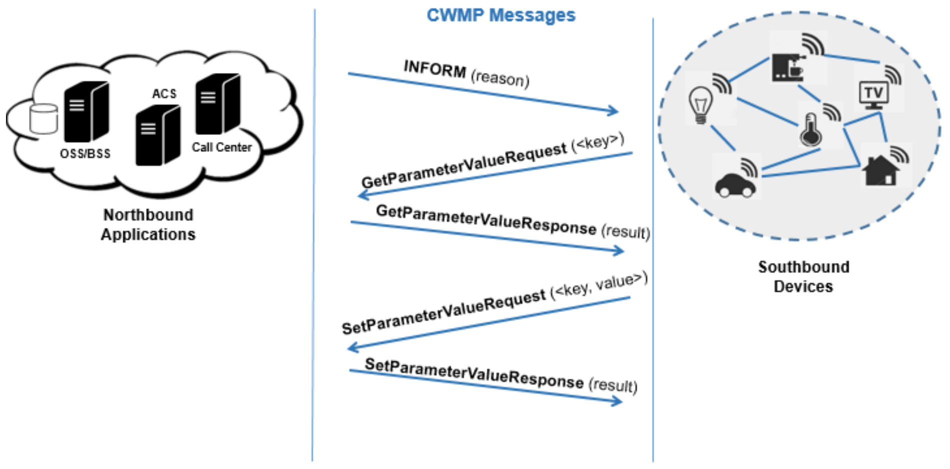

CWMP Messages
ACS
OSS/BSS
Call Center
Northbound Applications
INFORM (reason)
GetParameterValueRequest (<key>)
GetParameterValueResponse (result)
SetParameterValueRequest (<key, value>)
SetParameterValueResponse (result)
TV
Southbound Devices

FIGURE 2.19: Illustration of the CWMP Protocol messages.

After receiving all the answers or does not have requests, CPE closes the session. An Inform message initialize the management session. The client identifies this message, which is confirmed by an InformResponse message by the server. Subsequently, the client can request or assign one or more parameters with a GetParameterValue and SetParameterValue message. Both messages are committed with a SetParameterResponse or GetParameterResponse and the parameter values are updated. Finally, a management session is finalized.

Use Case

Incognito ACS is an integrated system to the SAC [96]. It allows to manage copyright of the subscribers, e.g., group of channels or videos on demand over IP. The SAC authorizes TR-069 gateway activation and diagnostics. In the gateway it is possible to execute commands to learn about devices, services, or customer quality. As another use case example, the COSMOS (CPE Operation Support Management and Optimization System) is a CWMP-based Operations Support System (OSS) used to provide integrated multi-function which has an easy to use operating environment. Multi-vendor CPEs (common gateways) are managed by COSMOS, and this system is described in [97].

2.5 IoT Device Management Platforms

The section addresses the most relevant IoT Device Management Platforms. The following solutions are considered: Management for the Internet of Things (ManIoT), Fiware, ONEM2M, SmartThings, RestThing, Xively, and Carriots.

2.5.1 Management for the Internet of Things (ManIoT)

ManIoT platform allows managing devices that make up the IoT environments [98]. Figure 2.20 shows the applications and sensors installed physically on IoT management environment.

The ManIoT platform takes into account the devices heterogeneity or "things". Therefore, ManIoT does not require modifications or installations of additional software on devices or applications in user devices. ManIoT accesses the applications through a Web user interface.

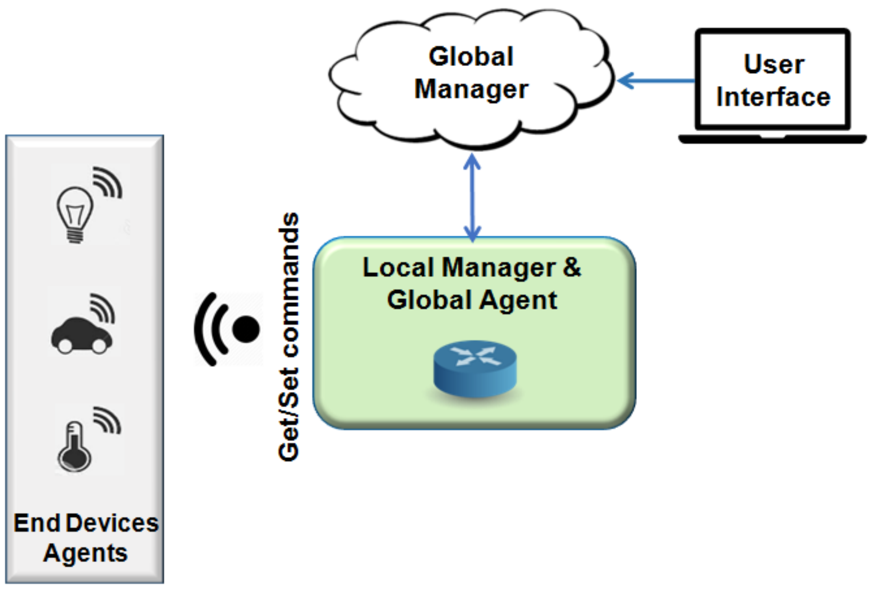


FIGURE 2.20: Illustration of the Management for the Internet of Things.

The ManIoT standardizes the data model and format used to applications, services, and devices communications. The device's status (on/off) and the Id (identification device) are characteristics used to model information. To integration with external systems, the platform uses popular protocols and data models of the industry, e.g, XML and RestFul API.

The ManIoT project has two management scopes, Local and Global/Remote. The Local Manager acts to control events performed by a user or application devices that make up a particular scenario, for example, turning a water valve on or off. The remote manager standardizes the actions by users in different scenarios, consumption rates in various areas defined by the water utility.

Local Management

The local manager acts based on information on the context within a scenario, i.e., the local manager can control and monitor the events, such as turning a lamp on or off.

The functions performed for each layer, as shown in Figure 2.21, are described below:

• **Application Layer:** The first layer consists of applications that use data provided by one or more devices, as well as platform services. Network users access applications through a web interface, and these applications, in turn, interact with ManIoT using function calls. Each application requests the platform to perform actions on the sensors based on

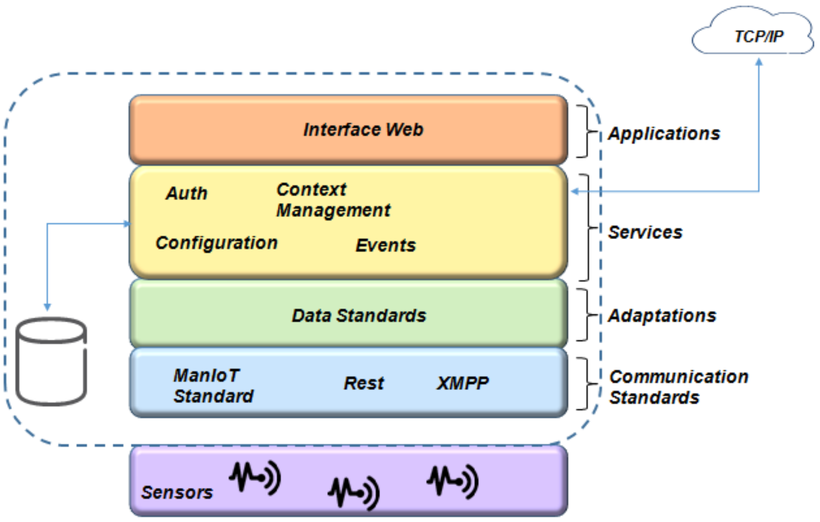


FIGURE 2.21: ManIoT: Local Management Architecture.

the implemented scenario, e.g., an energy management application requests turning an air conditioner on or off in order to reduce consumption.

• **Service Layer:** The second layer is formed by the services that support the applications and use the abstractions implemented by the drivers to communicate with the devices. Among the items in this layer are Storage, Scheduling, Authentication, Settings, Communication, Events, Conflict Management, and Context Management.

• **Adaptation Layer:** This layer is divided into two parts, the first one being responsible for standardizing the data and the second for dealing with the specificities of each device. Each device type has a specific driver that abstracts the specificities of access to its sensors and actuators, which allows the management of the services in an integrated way.

• **Communication Layer:** The layer consists of the different device access protocols. As mentioned earlier, the network may consist of devices that can use different application protocols (i.e UPnP or proprietary protocol) and different networks (ZigBee, WiFi).

• **A Layer of Things/Devices:** The last layer has the “Things”. There are two devices type: the real devices and the virtual devices. The actual devices are sensors and physical actuators, e.g., an intelligent lamp (actuator), a pressure sensor (sensor). Virtual devices have already captured information from a server connected to a TCP/IP network, e.g., a calendar or email service, or a social networking server.

Global Management

The global manager seeks to standardize the actions performed in different scenarios. It has two layers: Application and Services layers, as shown in Figure 2.22

The global services have the functions as those development in the local scope, as shown in the second layer of Figure 2.21 and Figure 2.22. Global scope services handle larger data sets and provide support for more comprehensive applications. For example, in the

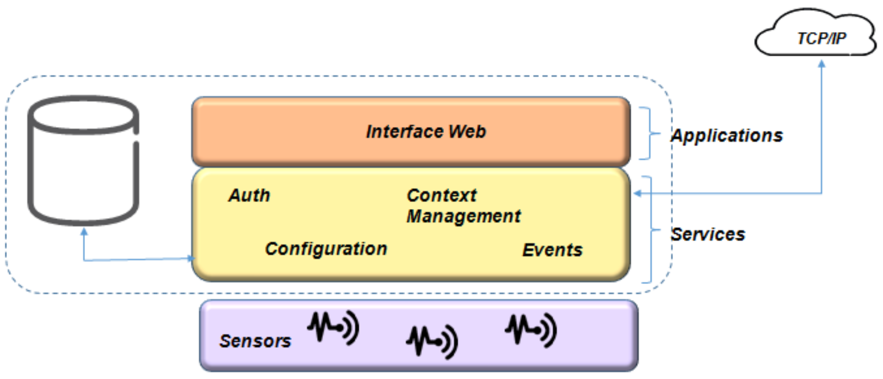


FIGURE 2.22: ManIoT: Global/Remote Management Architecture.

context of electrical management, the global manager must have the ability to manage possible power outages in several residences in a neighborhood. The actions defined by the global services are sent and executed in the devices of the respective local managers, using a TCP/IP connection.

Use Case

In the Intelligent Lighting Scenario, the lighting of an environment is adjusted with the presence of people and the existence of natural light. Bulbs are switched conform a person move in the room. Light intensity is inversely proportional to the amount of natural light.

The ManIoT prototype consumes approximately 0.05% of the bandwidth of these networks in the worst case [98]. These values are justified because of the small amount of data exchanged between the prototype of the local manager and the devices, thus reinforcing the minimal use of hardware resources.

There were results obtained with scenarios of intelligent lighting and automation of tasks using appropriate metrics to show the capacity of ManIoT to provide a dynamic adaptation and context science to the environment.

2.5.2 Fiware

Fiware is an open cloud platform, illustrated in 2.23, under development and created in an European FP-7 Project to support future Internet. Considered important to several areas, the Fiware has a set of generic enablers (GEs) [99]. According to standard IoT-RA (Figure 1.1), only the member function is implemented in Fiware Technologies. The platform offers support to various management protocols and standards. It supports OMA NGSI9/10, OMA LWM2M, MQTT, CoAP, and IPv6 [100]. The heterogeneous wireless networks have specific communication protocols to connected devices. Different data encodings make it difficult to find a global deployment.

The platform, illustrated in Figure 2.23, has a modular architecture that supports several IoT protocols, in which modules are called IoT Agents.However, the integrators must determine the protocol that will be used to connect and select the IoT Agent correct. The IoT

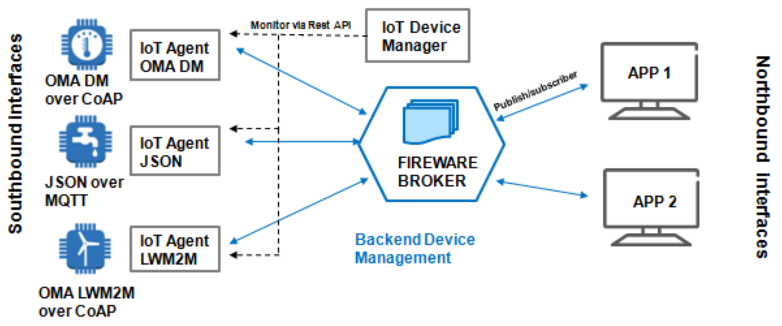


FIGURE 2.23: Illustration of the Fiware architecture.

Manager collects or sends data to devices that use heterogeneous protocols and translates them to a standard platform, simplifying the device management and integration [101].

Use Case

The Fiware project is used for orthopedics, podiatry, physiotherapists and related health services producing prosthesis. This work is time-consuming, cost-inefficient and causes many inconveniences to patients. The Ortholab aim is to produce advanced scan and manufacturing solutions to the insole sectors. With the Ortholab solution, orthopedists or physiotherapists will be able to take digital information of the patient's body part in an easy way and specify the parameters to 3D printers [102].

2.5.3 ONEM2M

ONEM2M was first released in 2015 and is a partnership project created to establish access-independent M2M service layers specifications. For the management protocol, it has its own technology called Device Management (DM) and it is also evaluating the possibility to implement OMA-DM, OMA-LWM2M or even CWMP [103] [104]. This platform has its own system and protocols, as described in [105].

Two basic types of entities make up the functional architecture of ONEM2M: AE (Application Entity) and CSE (Common Services Entity), as shown in Figure 2.24. Northbound and southbound connected devices are considered an AE. The AE needs to be aware of management data protocols or models. Device Management (DMG) enables device management capabilities in MNs (for example, M2M Gateways), ASNs, and ADNs (for example, M2M devices). Connected devices residing in an M2M network are managed by services provided by DMG. The information obtained from the AE is used for network administrative actions (e.g. diagnostics, troubleshooting) [106].

The Management Server/Client interface is the Mcc, which utilizes a device management technology (e.g. CWMP, OMA-DM, and LWM2M). Device management technology is used to manage the entities (MN, ASN or ADN, and DMG) and translates requests from other CSEs or from AEs to the device management technology. The Mcc interface is technology dependent, as above-described.

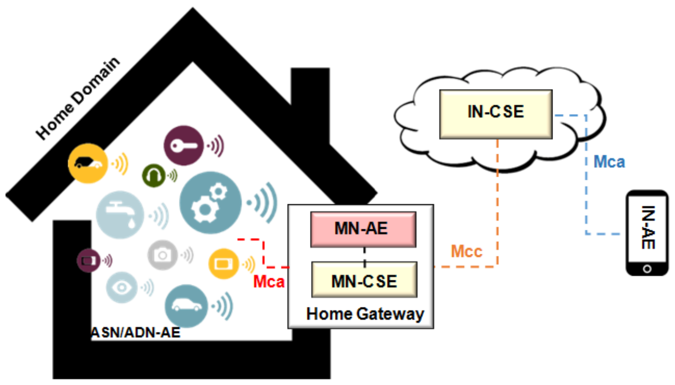


FIGURE 2.24: Illustration of the ONEM2M architecture.

Use Case

The home lighting use case [107] performs remote control of the lights in a home through a user's smartphone in the following manner: *i)* the lights are deployed and communicate with home gateway; *ii)* the home gateway communicates with the cloud platform, making it possible to control the lights remotely with the smartphone; *iii)* the cloud platform supports services to enable the smartphone to control the lights, e.g., discovery, data management, group management, publish/subscriber and others; *iv)* the user's smartphone hosts an application used to remotely control the lights, i.e., change light state (ON/OFF), discover available lights in the house, among other functions.

2.5.4 SmartThings

SmartThings is an open source solution used to build applications and connect with other devices. It allows new connected applications and supports applications (SmartApps) that communicate with other WebServices through RESTAPI. The SmartThings architecture illustrates the infrastructure blocks that interact with the devices shown in Figure 2.25. Communication of devices (sensors and actuators) with application is performed by the HUB entity. The messages are received, identified, and analyzed by a user device on the Device Handler. The response message is discriminated by JSON in SmartThings events. SmartApp handles devices through events managed by Subscription Management.

Use Case

The SmartThing project is used to optimize simple tasks in daily life. Between the functions used, the following were identified: presence sensors for security and light control, scheduling of house cleaning, and a sensor to get notifications when mail is received [108].

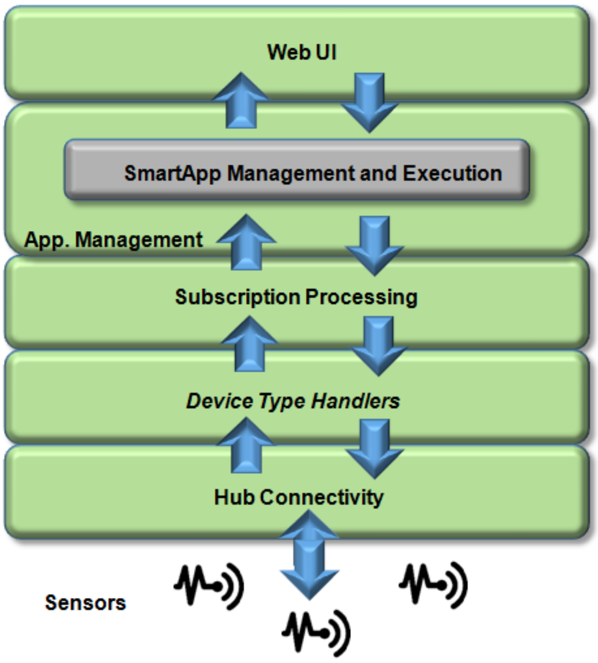


FIGURE 2.25: Illustration of the SmartThings architecture.

2.5.5 RestThing

The RestThing platform [109] is a Web service infrastructure based on REST with the purpose of hiding the devices heterogeneity and integrate devices into a network. This platform enables developers to build applications accessing physical and Web services, which are both manipulated by a REST-style interface.

As shown in Figure 2.26, the RestThing elements are: *i)* applications; *ii)* RESTful API; *iii)* service provider; *iv)* adaptation layer; *v)* embedded devices, and; *vi)* Web resources.

The RESTful API transmitted the data between sensors, gateways and Web applications using three types of data formats: JSON, XML, and CSV. For access to RESTful objects, the HTTP protocol operations used are: the GET method, used to retrieve the current device state; the PUT method, used to modify this device state; the POST method, used to create a new device; DELETE to remove a device, and, in addition, the LIST method, which allows all devices connected to the platform to be listed.

Use Case

The Monitor Temperature and Heartbeat application is the user interface in a smartphone that combines physical and Web resources in the Restful API. The real-time data view is used to obtain current data from WSNs. The smartphone updates this information by sending a GET to the gateway. The device number of the temperature sensor is what gets current internal lab room temperature as used in Smart Health environments [110].

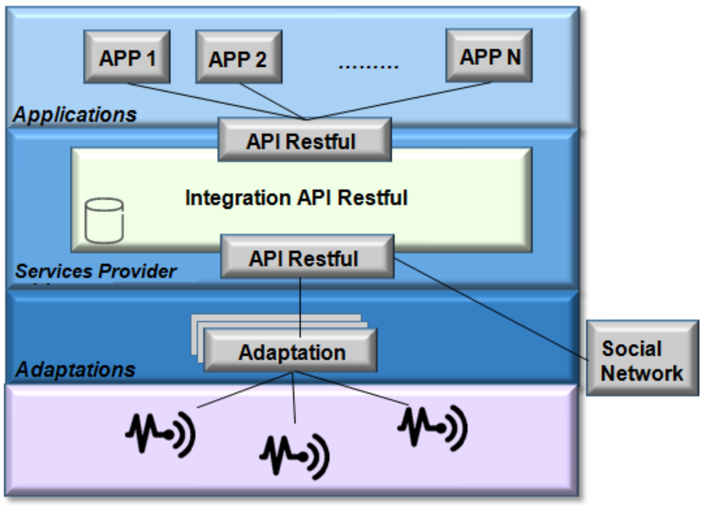


FIGURE 2.26: Illustration of the RestThings architecture.

2.5.6 Xively

Xively provides an API for managing data from the sensors/devices through cloud services. It allows historical data and provides events based on the data generated by sensors/devices. It was created based on the EcoDiF platform [31], Based on REST principles and Web standards such as HTTP and URIs. To minimize the incompatibility among different devices, the platform provides standardized interfaces. The data is organized into data points, streams, and feeds. A feed represents an environment data (i.e. a room) with its data streams, representing data sent by a particular sensor in that environment (i.e. temperatures of the monitored environment).

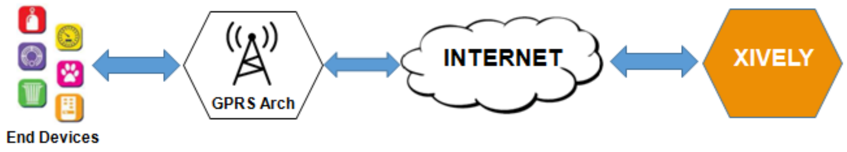


FIGURE 2.27: Illustration of the Xively architecture.

Xively is a commercial and closed source solution [111]. There are little details about the architecture of this platform, which is shown in Figure 2.27. The sensors send data to the platform in JSON, XML or CSV formats using the REST API, via sockets and through the MQTT protocol [112]. However, it is known that there are three ways to manage the devices connected to Xively. In the first case, through the methods implemented by HTTP, the GET method is used by a client program to retrieve data from a feed or data streams. The PUT method is used in the connected devices to send data. In the second case, a socket can be created to avoid the overhead of opening and closing HTTP communications under conditions in which too much data is exchanged. In addition, the first two cases allow the use of the SSL/TLS protocol to provide authentication and data encryption. Finally, the third case uses the publish and subscribe features of the MQTT protocol to send and retrieve data from devices.

Use Case

A scalable platform that addresses need residential customers and contractors were developed by SunStat Connect Thermostat, called Watts Water's SunTouch [113].

The Xively IoT Platform is used to power the remote connectivity of SunStat thermostats, which work with all SunTouch heating products. Furthermore, the Xively provides nearly instantaneous response times from the devices, while not sacrificing stability and reliability that consumers would expect from the heating control device. The Xively platform developed remote connectivity in the SunStat application, enabling consumers to control SunStat devices from a web browser or mobile device from anywhere in the world connected to the internet.

Another application is Blueprint [114], that consists of a scalable object directory model with a fast MQTT-based messaging broker. It has a secure provisioning process, which supports millions of secure connections between people, devices, and data around the world.

2.5.7 Carriots

Carriots is an IoT platform that manage data devices provided with cloud services, and that also connects devices to other devices and systems [115]. Therefore, if a system is connected to the platform, it can also be modeled as a device. From its RESTful API, Carriots aim to collect and store any data originated from the most diverse devices. The application engine can guarantee availability to its users no matter the volume of connected devices. These connected devices are associated with services (i.e., physical devices or other resources) and all services belong to a project. As shown in Figure 2.28, the logical architecture of Carriots consists of the following modules: *i)* Drivers Communication Services (REST API and External Communication Module) *ii)* Tracking (Control Panel, Logs, and Debug) *iii)* Interventions (Security, Project, and Device Management) *iv)* Monitoring (Business Rules and Event Processing with Big Data)

Data exchanged between devices, connected systems, and the platform can be represented in two different ways: *i)* the sensors send data in JSON or XML formats (in a particular platform format) using the REST API, or *ii)* through the MQTT message protocol [116].

The Project and Device Management module contains the projects created by users and provides device and its software management, i.e. device provisioning, enabling and disabling devices, and updating firmware. Storing and executing events in the form of scripts created using the Groovy programming language and using if-then-else rules is the responsibility of the Business Rules and Event Processing module.

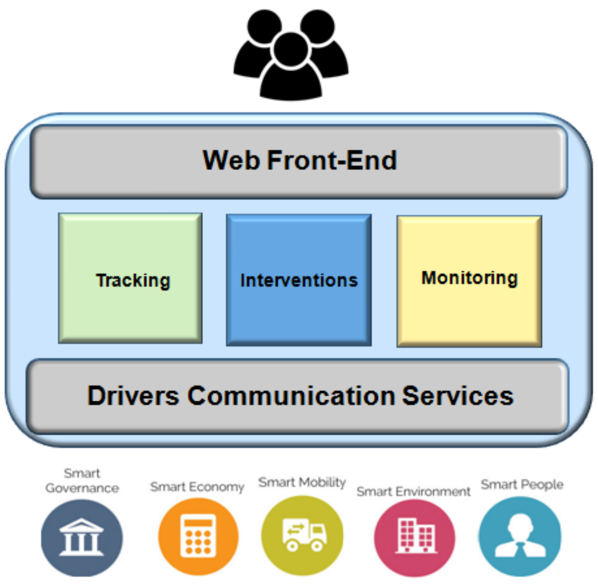


FIGURE 2.28: Illustration of the Carriots architecture.

Use Case

Nowadays, cities have the challenge of improving, protecting the environment, decreasing energy use, and reducing CO_2 emissions, as well as having to detect and correct any excesses in light consumption or in water spillage and also control expenses.

Carriots City Life is an IoT platform that works like the city brain. It collects data from different sensors or information reported by the citizens and mixes it all to better manage municipal services [117]. It's a cloud platform that allows people to collect, integrate, store and analyze all the city data with a global vision.

2.6 Performance Comparison, Analysis, and Open Issues

Main characteristics are based on studied management solutions. A qualitative study is performed in order to characterize management types and technologies used in the most relevant management protocols and platforms. Tables 2.5 and 2.6 summarize the main protocols characteristics considering their standardization resources, data, transport stacks, among others.

IoT network management protocols (shown in Table 2.5) were originated by the Internet Engineering Task-Force (IETF) for management of connected devices. The SNMP protocol was the basis for the creation of other protocols such as IoT-PIC and LNMP. Its simplicity in data modeling makes it a fast and a simple configuration protocol. NETCONF was created to be the successor of SNMP using the XML standard for request and response messages. The secure connection transport on SNMP is relevant because of its ease configuration in some scenarios. With the emergence of Software-Defined Networks (SDNs), new network

TABLE 2.5: Main characteristics comparison of the IoT network management protocols.

	SNMP/ LNMP	NETCONF	IoT-PIC/ XMPP	OVSDB	IEEE 1905
Standard	IETF	IETF	IETF	IETF	IEEE
Resource	OIDs	Paths	URLs	URLs	URLs
Data Modeling	SMI	YANG	WSDL	JSON	WSDL
Encoding	BER	XML	XML	JSON	XML
Transport Stack	UDP	SSH/TCP	HTTP, Web API	HTTP, SSL/TLS	HTTP, Web API

management protocols have been proposed, such as OVSDB. It uses JSON technology to expose its devices and network data for the systematic integration of applications.

TABLE 2.6: Main characteristics comparison of the IoT device management protocols.

	CoAP	OMA DM	OMA L2M2M	TMP	CWMP
Standard	IETF	IETF	IETF	IETF	Broadband Forum
Resource	URLs	URLs	URLs	URLs	URLs
Data Modeling	JSON	XML, JSON	XML, JSON	WSDL	XML WSDL
Encoding	JSON	XML, JSON	XML, JSON	XML	XML
Transport Stack	UDP HTTPS SSH/ TCP	UDP HTTPS SSH/ TCP	UDP HTTPS SSH/ TCP	HTTP Web API	HTTP SSH/ TCP

The IoT device management protocols (Table 2.6) were developed with Internet standards by the IETF. The Transport Layer Security (TLS) protocol and Secure Sockets Layer (SSL) protocol are used to secure transport of the information in the network as HTTPS and SSH protocols. In reference to the data modeling and encoding, the OVSDB and COMAN protocols use current Web technologies, such as XML and JSON, that expose network and device information to access other applications through an user name and password.

Table 2.7 summarizes a comparison among the most relevant IoT network management platforms, considering IMPReSS, OpenNMS, OpenDaylight, and Zabbix. Table 2.8 summarizes a comparison among the most relevant IoT device management platforms, considering Xively, ONEM2M, ManIoT, and other important solutions, regarding several important protocols and types of management approaches.

TABLE 2.7: Main characteristics comparison of the IoT network management platforms.

	IMPReSS	OpenNMS	OpenDaylight	Zabbix
SNMP		X	X	X
NETCONF		X	X	
IoT-PIC/XMPP	X		X	X
LNMP		X		X
OVSDB			X	
IEEE 1905				

All the platforms have Web user-interfaces and open source technologies, except IMPReSS (which was finalized in 2016). OpenDaylight supports more IoT network management protocols compared to other previously researched solutions. Zabbix is a popular platform for monitoring and management networks and differ in XMPP and NETCONF protocol with OpenNMS.

Almost all the IoT devices management platforms solved the heterogeneity and security problem.

No IoT device management platform meets all requirements. Some platforms have a range of features, but requirements such as interoperability are developed in different ways on each platform.

Despite attending to most requirements, SmartThings believe that supporting widely used protocols and Web technologies is enough to mitigate the problem of heterogeneity devices. However, support for other protocols is an important requirement for Carriots and Xively platforms. Other requirements (like context-awareness and dynamic adaptation) are barely discussed. Context-awareness is an approach to the inclusion of semantic data in a platform, e.g, location and collection time.

Platforms and new solutions are based on the SNMP architecture protocol, as shown in [120] [121], such as the use of the OVSDB protocol for the SDN architecture and the XMPP protocol shown in IoT-PIC. SNMP uses more efficient resources thus responding to a processing request up to ten times faster than NETCONF according to the study presented in [5] [122] [123]. The OpenDayLight modular platform supports more network protocols for IoT, such as SNMP, NETCONF, IoT-PIC/XMPP, OVSDB, and additional features. Then, it can be considered the most promising platform among the studied IoT network management platforms.

Based on this study, it is concluded that IoT device management is still in an early stage. The requirements have not been completely explored. Nevertheless, ManIoT and ONEM2M can be considered the best open source solutions, though ONEM2M supports a wider range of management protocols. Xively is the best proprietary platform according to the mapped characteristics. No solution can cover all the requirements of an RA. Thus, there are open research issues due to divergences between the technologies and approach researched.

TABLE 2.8: Main characteristics comparison between available device management platforms for IoT.

	YOAPY [118]	EcoDIF [31]	RestThing	SmartThings	IFTTT [119]	ManIoT	Xively	Carriots
OpenSource		X	X	X	X	X		
Heterogeneity		X	X	X	X	X	X	X
Security and Privacy		X		X	X	X	X	X
Scalability and Reliability		X	X	X	X	X	X	X
RFID	X		X			X	X	X
SNMP								X
NETCONF						X		
LNMP								
OVSDB								
IoT-PIC/XMPP					X	X	X	X
OMA-DM	X			X	X		X	
OMA-L2M2M	X			X	X		X	
TMP								
6LoWPAN	X					X	X	X
SOA		X	X	X			X	X
Data Management	X						X	X
Device Management		X	X	X	X	X	X	X
Local Management	X			X		X	X	X
Global Management	X	X	X	X	X	X	X	X
Remote Management	X	X	X	X	X	X	X	X
Context Management					X	X	X	X

2.6.1 Open research issues on IoT management

After the previous discussion regarding the available management protocols and platforms for IoT, this section identifies open research issues on IoT resources management. They are presented as follows:

1. Performance evaluation metrics: other performance metrics may be considered for performance comparison of the available solutions and others that may be proposed by the community. Thus, a comparative study using error probability, mean response time, and latency may be necessary to evaluate the performance of a given solution.

2. Energy saving: extending the lifespan of IoT applications in a network is a matter that may be considered. Connected devices have limited capabilities and should not be overloaded with high throughput. Thus, there is a need to use simpler and smaller packets on the network while not overlooking device security.

3. Security: it is a key issue to favor the advancement of IoT. However, due to devices heterogeneity, each company uses different security protocols. Therefore, standardization is a security related challenge. Also, as the importance and popularity of IoT devices increases and people provide more information on the topic, scammers and the most experienced cybercriminals continue to search new ways to attack and compromise a set of devices. Consequently, research related to this subject is on the rise in the research community.

4. Real-time management: various types of IoT application domains require high network availability. Health applications, for example, have a critical degree of availability and, therefore, require real-time management. However, limited device resources and power savings are related features for a management solution that should not be overhead in its communication within the network.

5. Interoperability: There are currently several types of devices, protocols, and communication technologies that determine the heterogeneity of an IoT network. These devices must communicate and inter-operate to provide a network service to users. Some researches assume that new gateways developed must support several protocols. Furthermore, there is still research on the low power device network as seen in [124].

6. Scalability: The number of devices connected to the network increases exponentially and, thus, the scalability of an IoT network is a critical requirement. The solutions found in literature do not address features regarding scalability. With this, IoT network management must support scalability due to the great evolution of IoT devices and related technologies.

2.7 Summary

The resources in terms of scalability, interoperability, security, energy savings, and IoT solutions (protocols and platforms) were studied to select the best solution to be evaluated. IoT network solutions continue being based on the SNMP protocol. The NETCONF protocol was developed to be the natural successor of SNMP since SNMP is focused on monitoring and is desired regarding management. The support for each platform tries to improve latency, heterogeneity, scalability, and robustness. The OpenDayLight platform can be considered the best solution based on the supported protocols, and characteristics described.

This chapter gave an insight to determine the best choice of network management protocols and IoT platforms for an IoT management solution to depoy in real environments.

A detailed description and discussion of the topic was performed. Other research issues on the topic were also been identified.

Chapter 3

Performance Evaluation of IoT Network Management Platforms and Protocols

This chapter focuses on a performance evaluation study of IoT protocols and network management platforms available in the literature. It presents an extensive performance study to support a decision regarding the most relevant protocol and platform to share information with the connected devices. These chosen technologies should integrate the IoT management platform proposed in this work.

3.1 Experimental Scenario

The typical IoT application architecture can consider three layers. The gateway connect sensor networks with traditional communication networks creating heterogeneous networks. The security problems with sensor networks, mobile communications network, and the Internet are more particular in IoT networks due to privacy protection problem.

In the proposed system, the real environment should emulate an IoT smart light system, where IoT network management protocols and platforms should be evaluated according to quantitative and qualitative metrics described below, e.g, request/response time average, throughput, data rate, security platforms, and traffic manager into server-side to determinate the standard and most promising technologies to develop.

3.1.1 Hardware and Software Specifications

Generally, a sensor node needs low CPU consumption and energy-saving because it is a restricted device in terms of hardware and software requirements. Encrypt and decrypt programs cannot use large storage and high-power. So Encryption mechanism in IoT should be lightweight [62]. New operating systems are emerging to address requirements, such as TinyOS [125] and Contiki OS [126]. In Table 3.1, the system requirements used to IoT smart light system are described.

The high processors must be concentrated on service servers and applications. Platform servers are responsible for managing this processing, in addition to the safe manipulation

TABLE 3.1: System Requirements of Hardware and Software to create the experimental scenario.

System Requirements	Platform Servers	Sensors	Gateway
CPU	Intel Core 2 Duo	32-bit ARM Cortex-M3	32-bit ARM Cortex-M3
Clock Speed	3.00 GHz	48 MHz	32 MHz
RAM	2GB	128KB	512KB, 256KB or 128KB
Operational System	Ubuntu 14.04 LTS	Contiki OS	Contiki OS
Java SDK	Java 7	-	-
Security Protocol	SSL MD5	AES-128	AES-128 256-SHA2
Communication Technologies	IPv4 Ethernet	6LoWPAN, ZigBee Bluetooth LE	ZigBee

and availability of devices information. Therefore, the system requirements of these platforms must be greater than devices.

3.1.2 System Architecture

This section describes the proposed scenario and the configuration of the real environment used to perform the experiments and collect the corresponding results. This model is important to evaluate the performance of the IoT network management protocols under study (SNMP and NETCONF). The real environment deployed to evaluate the performance of the management protocols includes the following equipment: (*i*) Web platform integrated with OpenDayLight, (*ii*) a gateway and, (*iii*) light, temperature, and humidity sensors.

This IoT network management environment, shown in Figures 3.1 and 3.2, includes a responsive user interface using the bootstrap framework for the front-end. The back-end of the application is based on the OpenDayLight platform, together, with PHP language. OpenDayLight provides the RESTful Web service to manage the network elements through NETCONF and SNMP protocols supported by the platform. The network follows a common Internet Engineering Task-Force (IETF) layered architecture for wireless sensor networks [11]. It uses the standard IEEE 802.15.4 as physical and data link layers, Internet Protocol version 6 (IPv6) using 6LoWPAN for identification and sensors connection with the gateway, IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) as routing protocol, and the Constrained Application Protocol (CoAP) as the application layer protocol for management and configuration of the sensor nodes. Concerning security, encryption is using 128-bit AES key.

The sensors use the Contiki operating system (OS) [126], and Momote components (using the CC2650 microcontroller) [127] integrated with a low-power 2.4GHz multi-protocol

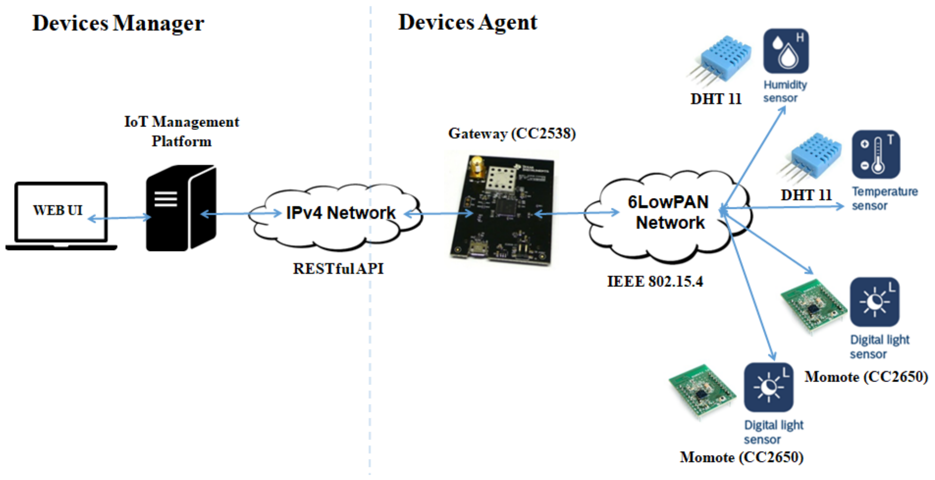


FIGURE 3.1: Illustration of the real environment system architecture used to perform the experiments.

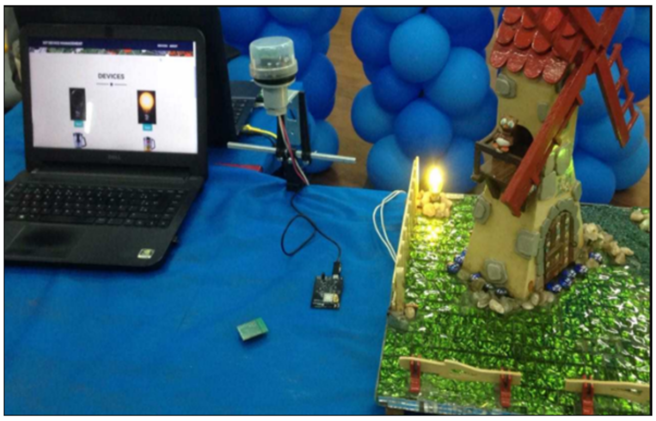


FIGURE 3.2: Photography of the real environment scenario used to perform the experiments.

radio. The DHT11 sensor monitors the environmental temperature, humidity, and brightness (light). The gateway uses the CC2538 microcontroller [128] from Texas Instruments. It is used as personal area network (PAN) coordinator and responsible for channeling the messages from the star topology network to the external network, through a microcontroller that also runs the Contiki OS.

Figure 3.3 shows the layered architecture of the deployed solution. As may be seen, users interact with the system and perform management of the network elements through the services supported by OpenDayLight (ODL).

SNMP or NETCONF protocol is a northbound or southbound device manager that is capable of being connected remotely to the device's agent. This device agent exposes its configuration/operational information for monitoring and controlling one or more device managers (using SNMP or NETCONF). The device managers (running in the background at the server) obtain or control device agent resources and make the information available

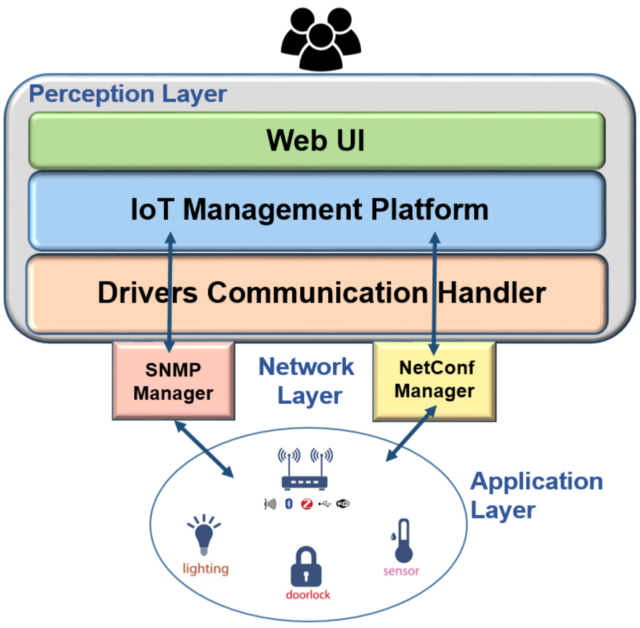


FIGURE 3.3: Illustration of the layered architecture used in the real environment.

to ODL through a driver handler that aims to translate various IoT network protocol communication patterns into the RESTful API format supported by ODL.

Figure 3.4 shows the layered architecture of the deployed solution. The application layer includes a responsive user interface using the bootstrap with ODL and OpenNMS platform. Both IoT management platforms provide a RESTful Web service to manage the equipment through NETCONF and SNMP.

The IoT management protocols, i.e., SNMP and NETCONF, are loaded in a northbound or southbound device's agent that is capable of being managed remotely to one or more devices' manager. This device agent exposes its configuration/operational information for monitoring and controlling by device managers. The device managers (running in background at the server) obtain device agent resources and turn available the information to platforms through a driver handler that aims to translate various IoT network and devices protocol communication patterns into the RESTful API format supported by ODL and OpenNMS platform.

3.2 Qualitative Metrics

The traditional qualitative evaluation has certain similar features of the IoT management. The Table 3.2 contains the features and a detailed description of each metric used to evaluate protocols and platforms. These metrics goal to reduce subjectivity and help to select the promising protocols and platforms to evaluate in a quantitative performance evaluation study.

TABLE 3.2: Qualitative metrics description for IoT Management Protocols.

Qualitative Metrics	Brief Description
FCAPS Model	Indicate the layers of the network management model: fault management, configuration management, accounting management, performance management, and security management.
Popularity	The popularity idea is used to measure how well the protocol is known, disseminated in research and commercially.
Supported standard	The protocols that can be used to communicate with other protocols and they are mandatory to ensure the interoperability among devices like JSON, XML, and CoAP.
Security Feature	Security is a key issue on IoT and has the intention to increase assume that device credentials were not compromised.
Maturity	The maturity metric represents the maturity of the development and the amount of users or equipment using this technology.
IPv6 Support	IPv6 allows more users and devices to communicate on the Internet by using bigger numbers to create IP addresses.
Models Based	Model-Based Design (MBD) is a mathematical and visual method of addressing problems associated with designing complex control, signal processing, and communication systems

3.2.1 IoT Network Management Protocols

Qualitative analysis of evaluated protocols presents difficulties due to the fact that there is no common accepted standard by which comparisons can be performed. The proposed evaluation uses a number of criteria that intuitively come into mind and are under use in the topic. Security is a key aspect on IoT management for all systems, due the high quantity of the devices that can be vulnerable to intrusions [129]. Model type and representation should have data model flexibility because the management supports a variety of physical and virtual networks [130] [131]. Thus, the metrics proposed measured a protocol with more flexibility, security, and maturity to perform the evaluation. Table 3.3 summarizes the application of proposed qualitative metrics to evaluate management protocols considered in this study and are described below.

FCAPS Model

FCAPS is an acronym to indicate the layers of the network management model: i) fault management, ii) configuration management, iii) accounting management, iv) performance management, and v) security management. Networks faults and problems are found and fixed in the fault management. In the configuration management, the network is monitored and controlled, keeping track of hardware and software modifications. For accounting management, network resources are distributed and departments are charged for their end users network usage, such as long-distance or bandwidth usage per user. Network congestion and bottlenecks are minimized in the performance management. Security management applies equally to both outside intruders and internal users; not all network hackers are from outside

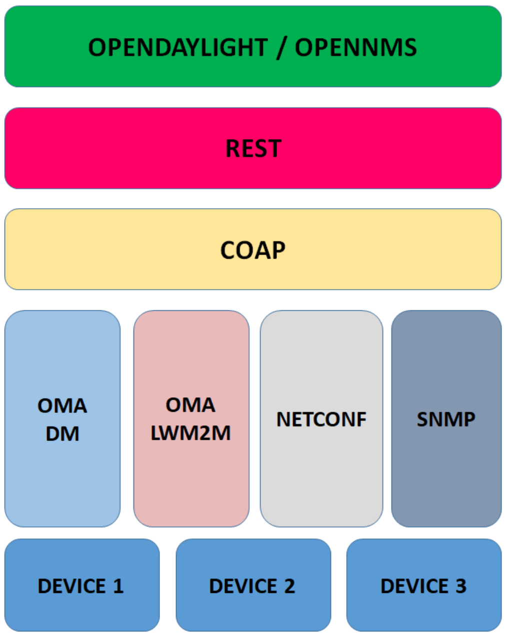


FIGURE 3.4: Illustration of the platforms layered architecture.

TABLE 3.3: Qualitative performance evaluation of the IoT Network Management Protocols considered in this study.

Qualitative Metrics	SNMP/ LNMP	NetConf	OVSDB	IoT-PIC	IEEE 1905
FCAPS Model	Yes	Yes	Yes	Yes	Yes
Popularity	Highest	Highest	High	Low	Low
Supported standard	SMI	YANG	JSON	WSDL	WSDL
Security feature	Yes	Yes	Yes	Partial	Partial
Maturity	Highest	Highest	High	Low	Low
IPv6 Support	Yes	Yes	Yes	Yes	Yes
Models-Based	MIB	Yang	Open vSwitch	XML	ALME

an organization.

Popularity

The popularity idea is abstract because attempting to measure how well the protocol is known, accepted, and disseminated in research and commercially. An alternative would be to perform a manual Google search using the name of each middleware platform and manually check if the first results are relevant (according to [132], 91% of users do not go beyond the first three results). To measure popularity is generally to execute the manual Google search, but exists same tools that help to measure this popularity, i.e., MOZ [133] and Alexa Page Rank [134]. The Google search considers the user location and previous browser history when returning results. Thus, the popularity measure was performed through the Google search with cookies and history of browser cleans.

Supported standard

The protocols that can be used to communicate with other protocols to maximize compatibility with devices (i.e., application layer) are listed. The IETF determines some standard protocols to use in commercial equipment. These standards are mandatory to ensure the interoperability among devices like JSON, XML, and CoAP.

Security feature

Security and privacy is an essentials aspect of any system, and it seems IoT developers are relegating it to the second plan, so products can be developed faster. In this metric, it is assumed that IoT network protocol should ensure data security. Security is a key issue on IoT (as well as in ICT technologies) and has the intention to increase security and assume that device credentials were not compromised.

Maturity

Kerzner *et al.* [135] says that maturity can be defined as the development of systems and processes that are repetitive by nature and guarantee a high probability that each of them is a success. However, the author makes a restriction stating that repetitive systems and processes are not a guarantee of success, but increase the probability.

The maturity metric represents the maturity of the development and the amount of users or equipment using this technology. Development maturity is measured through Full Application Lifecycle Management (ALM) with continuous integration and test coverage. When there are more users or quantity equipment using the protocol or platform, it makes the system more mature due to the higher occurrence of bugs found and solved turning the version more stable.

IPv6 Support

IPv6 is a standard developed by the Internet Engineering Task Force (IETF), the organization responsible for all the Internet technologies. IETF, anticipating the need for more IP addresses, created IPv6 to accommodate the growing number of users and devices accessing the Internet. IPv6 is the Internet's next-generation protocol, designed to replace the current Internet Protocol, IP Version 4.

IPv6 allows more users and devices to communicate on the Internet by using bigger numbers to create IP addresses. Under IPv4, every IP address has 32 bits long, which allows 4.3 billion unique addresses. In comparison, IPv6 addresses have 128 bits, which allow for approximately three hundred and forty trillion unique IP addresses. IPv6 offers other networking advantages. In most cases, computers and applications will detect and take advantage of IPv6-enabled networks and services without requiring any action from the user. IPv6 also reduces the need for Network Address Translation, a service that allows multiple clients to share a single IP address, but it is not always reliable.

In this qualitative metric, the protocol should support complete, partial, or not of the IPv6 requirement.

Models-Based

The description languages help in the planning, systemic analysis, and resource specification of the described system. In general, there are tools that allow for the high level description of a system and its properties. The output of a description language is, most often, XML or XML-based [136] code that can be used as input in certain parsing software. Several network description languages (NDLs) exist in computer networks topic primarily focusing on physical rather than virtual networks [137].

The Network Description Language (NDL) [138] is the most prominent among the relevant proposals. It is based on the RDF language, a metadata data model, part of World Wide Web (W3C) consortium's specification for the theoretical description and modeling of Internet resources [139]. Network information is categorized in network topologies, technology layers, device configurations, capabilities, and topology aggregations. NDL allows comprehensive this network information. A Management Information Base (MIB) is a logical information store consolidating entity details, organized in a hierarchical (tree-structured) manner. Accessing a MIB involves using a Simple Network Management Protocol (SNMP). The current properties of the managed objects are populated into the MIB information store by means of specialized software (MIB module).

3.2.2 IoT Network Management Platforms

Based on platforms, the qualitative analysis proposes an evaluation in the following aspects: i) security, ii) availability, iii) scalability, and iv) robustness. The platforms evaluated are open source and provides a Web user interface as pre-condition to enter in this assessment.Security is an important metric due to the platform's store all the device information to possible monitoring and controls these devices. The Web application has availability and robustness principals that are important requirements for an IoT management environment. This platforms server should be robust and scalable to available the management services at all moments. Table 3.4 and 3.5 summarizes the proposed qualitative platforms metrics that are presented below.

TABLE 3.4: Qualitative performance evaluation of the IoT Network Management Platforms considered in this study.

Quantitative Metrics	IMPReSS	OpenNMS	OpenDaylight	Zabbix
SSL certificate	No	No	No	No
Security feature	No	Partial	Partial	No
S3P	No	Yes	Yes	Partial
Modular-Driven	No	Yes	No	No
Maturity	Low	High	High	High
Interface	IDE Based	Web	Web	Web
Auto-Discovery	No	Yes	Yes	Yes
IPv6 Support	No	Yes	Yes	Yes
NetFlow Support	No	No	Yes	No

TABLE 3.5: Supported network management protocols by IoT network platforms.

	IMPReSS	OpenNMS	OpenDaylight	Zabbix
SNMP	No	Yes	Yes	Yes
NetConf	No	Partial	Yes	No
IoT-PIC/XMPP	Yes	No	Yes	Partial
LNMP	No	Partial	No	No
OVSDB	No	No	Yes	No
IEEE 1905	No	No	No	No

Protocols Network Supports (GPL)

IoT management protocols are responsible for managing and monitoring devices and networks. Each platform has a number of supported devices and network management protocols. This metric considers only the open source network (GPL) protocols supported by the platforms and will be listed later.

Secure Socket Layer certificate

Secure Socket Layer (SSL) Certificate is used to protect important information from users browsing a Website, preventing it from being intercepted, captured or viewed while transferring data to the server hosting the application. This protection is created from a strong encryption key that shuffles the information sent by the user so that it is impossible to discover the content from within the key and the only place that can unscramble this content in the server where the SSL Certificate is installed. So any attempt to capture a data packet becomes irrelevant, even if someone is able to intercept the data, it will be impossible to read the content.

Security, Scalability, Stability and Performance

The ODL community provides continual improvements across all its projects in the areas of security, scalability, stability and performance, or "S3P". The testing and integration groups, along with people from each individual project, work together to run ongoing experiments that give developers real-time results to see how changes affect S3P. It is continuing to evolve the development process to ensure that one can understand and monitor improvements in each one of these four areas. ODL is also working with OPNFV supporting a Controller Performance Testing project that would create industry wide performance experiments for SDN controllers in realistic, large, and automated deployments.

Security is a key issue in focus by ODL. The platform provides a framework for Authentication, Authorization and Accounting (AAA), as well as automatic discovery and securing of network devices and controllers. AAA have a strong security team and process to respond to any vulnerabilities immediately. In general, open source software has major advantages when it comes to security: anyone can find and report vulnerabilities; one can draw on a wide array of experts and developers across companies to discuss and fix vulnerabilities; and the community-at-large can see how such issues are addressed transparently and understand if the issue really has been fixed.

Modular-Driven

This technique separate the functionality of the system through modules, where each contains the only necessary to execute the aspect of the desired functionality. The elements defined in the interface are detectable by other modules. With modular programming, concerns are separated, so modules perform logically discrete functions, interacting through well-defined interfaces. Typically, these are also compiled separately, through separate compilation and then linked by a linker. This facilitates the creation of smaller projects that make up the entire project, making development and maintenance easier to complete. The IoT management platforms have modular systems that absorb both network and device management.

Maturity

Described in the previous Section.

User Interface

The User Interface is a fundamental part of software. It is the part of the system visible to the user, through which, depending on its characteristics, it can be a useful tool to accomplish the tasks, or if poorly designed, can become a decisive point in rejecting a system. Current interfaces are intended to provide the most "user-friendly" peoples and computer interaction possible. In this way, it should be easy to use by the user, providing simple and consistent sequences of interaction, clearly showing the alternatives available at each step of the interaction without confusing or leaving the user unsafe. It must go unnoticed so that an user can only focus on the problem that he/she wants to solve using the system. Currently, the most commonly used Interfaces are the application interfaces for smartphone, but it is restricted to smartphones only, and the Web interfaces that can be accessed from any device with Internet connection.

Auto-Discovery

The Auto-Discovery requirement is used to discover new devices in the IoT environment and registration these devices in a network. The device is inserted in the IoT environment scope and this should be able to be discovered and registered within the system. These functionalities are the responsible of the IoT platforms that manage the devices. This requirement is determinant for an IoT scenario because the number of devices will grow exponentially and they do not need to be registered manually, the environment is responsible for this role.

IPv6 Support

Described in the previous Section.

NetFlow Support

For years, regular monitoring of a network was performed by the SNMP protocol that delivers an overview of the IT infrastructure; giving network administrators information about the availability of its components. Today, when the availability and proper working of a company's network is crucial for its existence, much more information and sophisticated methods are needed. To get this information, Cisco has created NetFlow, a standard for collecting network traffic statistics from routers, switches, or specialized network probes.

NetFlow provides information from layer 3 and layer 4 which means IP addresses, ports, protocol, timestamps, number of bytes, packets, flags, and several other technical details. It could be easily imagined as a list of phone calls. One knows who communicates with whom, when, how long, how often etc.; but people do not know what the subject of the conversation was. By collecting, storing, and analyzing this aggregated information, network administrators, security engineers, or operations managers can run proceed several tasks.

3.3 Quantitative Metrics

Qualitative analysis of examined platforms presents difficulties because there is not a commonly accepted standard by which comparisons can be performed. The proposed assessment approach uses some criteria such as (*i*) security, (*ii*) availability, (*iii*) scalability, and (*iv*) robustness. Table 3.6 summarizes the proposed quantitative metrics for the protocols and platforms evaluation.

TABLE 3.6: Quantitative metrics description for IoT Management Platforms.

Quantitative Metrics	Brief Description
Packet Size	Packet size sent from the source to the destination.
Latency	This metric focuses on latency of request / response messages to IoT management, in seconds.
Throughput	Throughput is the rate of successful message delivery over a communication channel.
Average response time	It is the average response time (ms) of the IoT platform, especially in high load scenarios is crucial
Error percentage	The server can deal with the incoming load without experimenting errors

3.3.1 IoT Network Management Platforms and Protocols

The conditions for a fair comparison among the platforms should consider the same characteristics across all the studied aspects. The platforms are purely software, that means that hosting machine will have the same available resources (memory, processing power, disk space, etc.). In practice, this means that, with more resources, the local instance can perform better in comparison with fewer resources. Quantitative comparisons are easily translated into numbers and graphs. They are describe below.

The quantitative metrics proposed to evaluate the performance of management platforms are the following: (*i*) packet size, (*ii*) error percentage, (*iii*) variation of response times, (*iv*) latency, and (*v*) throughput.

Packet Size
The packet size can even aid in load balancing of the scenario. The devices are using IEEE 802.15.4, which supports transfer rates between 20 and 250 Kbps [140]. Most of the energy consumption in devices is due to communication. Therefore, knowing the packet size that is necessary to communicate, devices can better manage critical resources such as battery level and, in advanced cases, they can even plan the time intervals to transmit. Analyzing packet size, it is important to remember that REST communications usually have a response message that should also be accounted.

Latency
The challenge with IoT devices is that they run on batteries. In order to preserve the lifespan of batteries, IoT devices systematically wake up from sleep mode to retrieve new information. The longer the device is asleep, the less power it consumes. This also means that there are fewer opportunities for information to be exchanged. This impacts the performance of the device, causing it running slower (known as latency). In today's devices, low power consumption and low latency are in conflict with one another. Because of the advent of IoT technologies, however, finding a low-power and a low-latency solution is of the utmost importance.

Throughput
Throughput refers to the performance of tasks by a computing service or device over a specific period. It measures the amount of completed work against time consumed and may be used to measure the performance by a number of simultaneous users and application/system responsiveness. Similarly, for network communications, throughput in IoT networks is measured by calculating the amount of data transferred between locations during a specified period, generally resulting as kilobytes per second (KBps), megabytes per second (MBps), and gigabytes per second (Gbps).

Average response time
Response time refers to the amount of time that a software needs to process information. This average time is calculated with request and response time among IoT management platform and connected devices. Response time can be critical depending on the scenario. In this case, response time can be the round-trip time (RTT), in milliseconds. In high load scenarios, where a significant amount of data is sent to a server, the response time is crucial.

Error percentage
IoT management platforms will eventually handle a high amount of data due to the exponential growth of the connected devices. In a real scenario, many devices requests will be forwarded to the platforms that will be processed and sends response messages to devices. These platforms should be robust and always available. Resending those requests consume

much energy in devices and overload of the network causing congesting. A viability criterion stating the maximum tolerated error percentage should be established according to the proposed scenario.

3.4 Comparison Evaluation of Protocols and Platforms

Based on the main characteristics of the studied management solutions, a qualitative study is performed in order to characterize management types and technologies used in the most relevant management protocols and platforms. The experimental scenarios were configured with 10,000 users requesting management of a real environment of a smart lighting system using the following metrics: throughput, latency, error percentage, packet size, average response time, and others were also evaluated. The *GET* function is a message to request information from the devices. In the experiments, the users use OpenDayLight and OpenNMS to request the GET messages with the startup time of 1s, 5s, 8s, 10s, and 15s, respectively, for SNMP and NETCONF protocols obtaining results through the Apache JMeter Tool [141]. The corresponding results were collected and compared in order to determine the most viable platform in different protocols for managing IoT networks.

Figure 3.5 presents the percentage of users request errors occurred for different time intervals using ODL and OpenNMS platforms.

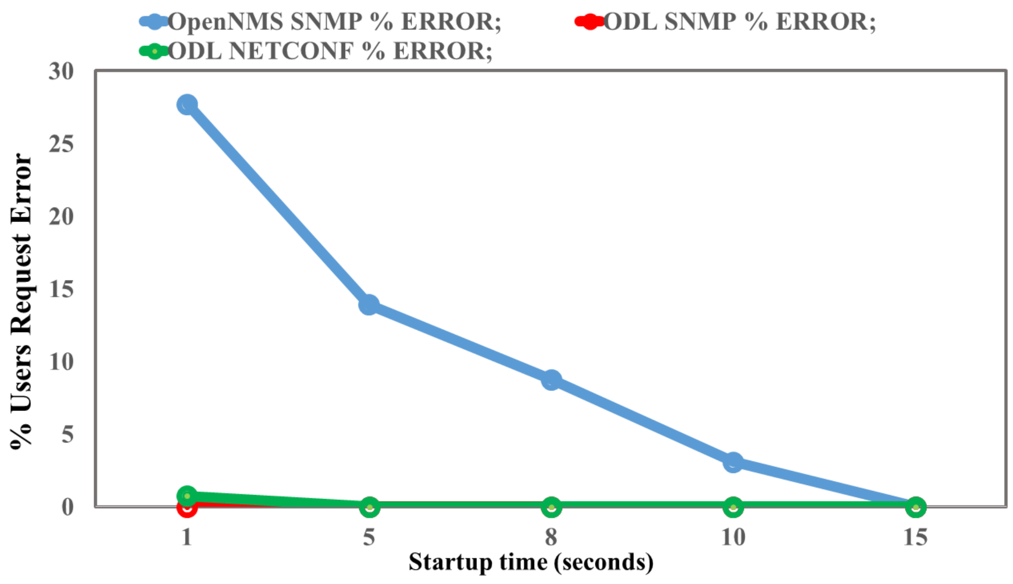


FIGURE 3.5: Error percentage of NETCONF and SNMP GET request with 10,000 Users for ODL and OpenNMS.

In ODL platform, it is observed that NETCONF and SNMP protocols, practically, do not have GET request errors in all the considered startup times. Nevertheless, OpenNMS obtain GET request errors in all the experiments. Thus, ODL is an IoT management platform with a better performance evaluation in comparison to OpenNMS. The relevant reason for OpenNMS having more request errors is the fact this platform has more GET messages collisions between users when they perform the request with any time interval.

Figure 3.6 presents the latency average histogram occurred for all GET request using ODL and OpenNMS platforms.

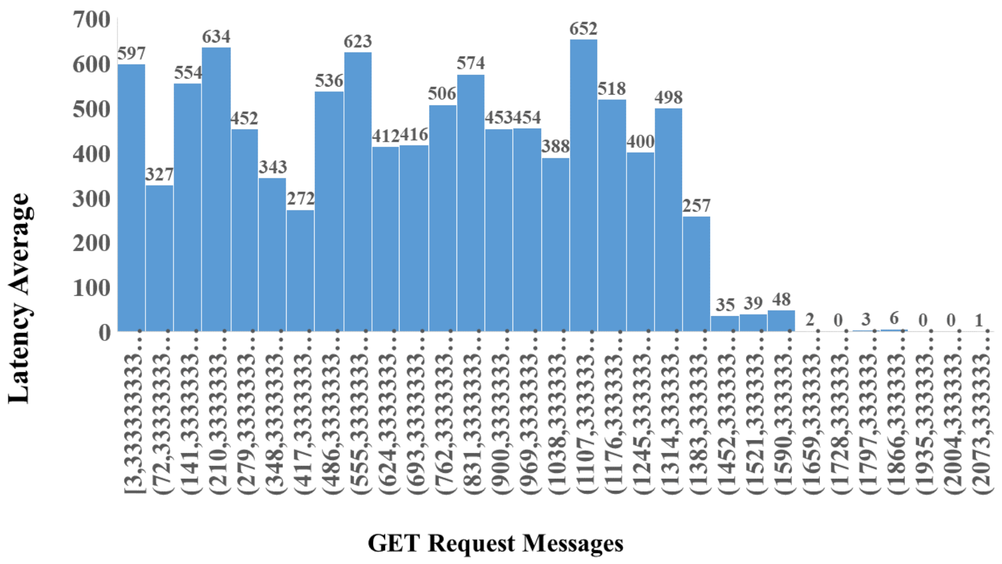


FIGURE 3.6: Latency average histogram of NETCONF and SNMP GET request with 10,000 Users for ODL and OpenNMS.

In summary, the latency average metric was measured with an average of all GET requests for the SNMP and NETCONF protocols on both platforms. It is observed that the latency result is approximately of 7 and 10 minutes. For IoT real-time and health scenarios this value for device management is impractical due to 30% of the GET request with high latency.

Figure 3.7 shows the results of the experiments considering the user average response time (in milliseconds) at different time intervals for the platforms under study. This evaluation should be based on the first scenario because the IoT environment needs devices and network management with lowest response time. In OpenDayLight, NETCONF protocol has an average response time of 3 seconds while the SNMP protocol has an average of 9 seconds, both for 10,000 users. The OpenNMS supports only SNMP that has an average response of 5 seconds. Like this scenario, NETCONF obtains an average response time better than SNMP in ODL and OpenNMS platforms. Therefore, NETCONF is a robust protocol that does have the desired performance to real-time environments due to its low average response time.

Table 3.7 measures the packet size of the GET messages NETCONF and SNMP using ODL and OpenNMS platforms. Using the Wireshark tool, the packet ratio of the IoT platforms were evaluated. It should be performed only by SNMP because NETCONF is not a native protocol in OpenNMS. In SNMP, there was a difference of the 78 bytes in ODL packet ratio regarding OpenNMS. It may conclude that ODL platform can deliver the same

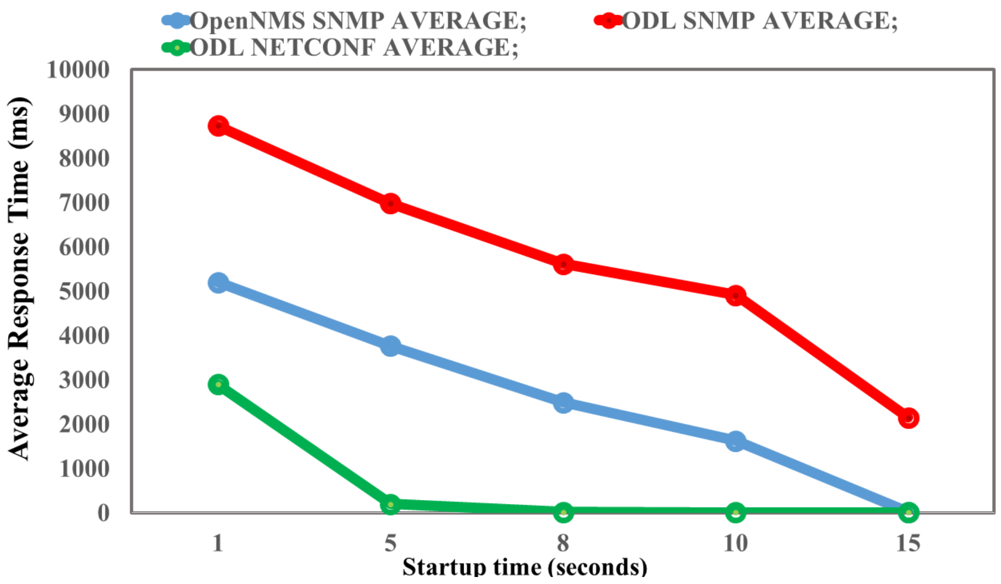


FIGURE 3.7: NETCONF and SNMP GET average response time (in milliseconds) from 10,000 Users for ODL and OpenNMS.

requests with less loss of packets and energy-saving with a package about 50% smaller than OpenNMS.

TABLE 3.7: NETCONF and SNMP GET packet message ratio for ODL and OpenNMS.

IoT Network Management Protocol	OpenDayLight	OpenNMS
NETCONF	66 bytes	-
SNMP	86 bytes	144 bytes

Figure 3.8 presents the throughput of user requests occurred for different time intervals using NETCONF and SNMP network management protocols through ODL and OpenNMS platforms under evaluation.

The ODL presents more throughput in comparison with OpenNMS, but this was already expected because ODL is a more robust platform. For SNMP evaluation in both platforms, OpenNMS achieved a performance around 300 Kbps more than the ODL. In more cases, the throughput information is directly related to the probability of error management due to the higher amount of throughput, the user average decreases, thus reducing the percentage of error in GET request for management of the network device. However, in this scenario, OpenNMS shows higher throughput, but there was more request error percentage because the platform is not robust and does not guarantee messages delivery.

Based on the obtained results, it may be concluded that OpenDayLight platform is viable for IoT network and devices management in IoT environments with the need to manage large amounts of data in real-time management with energy savings. On the other hand, it

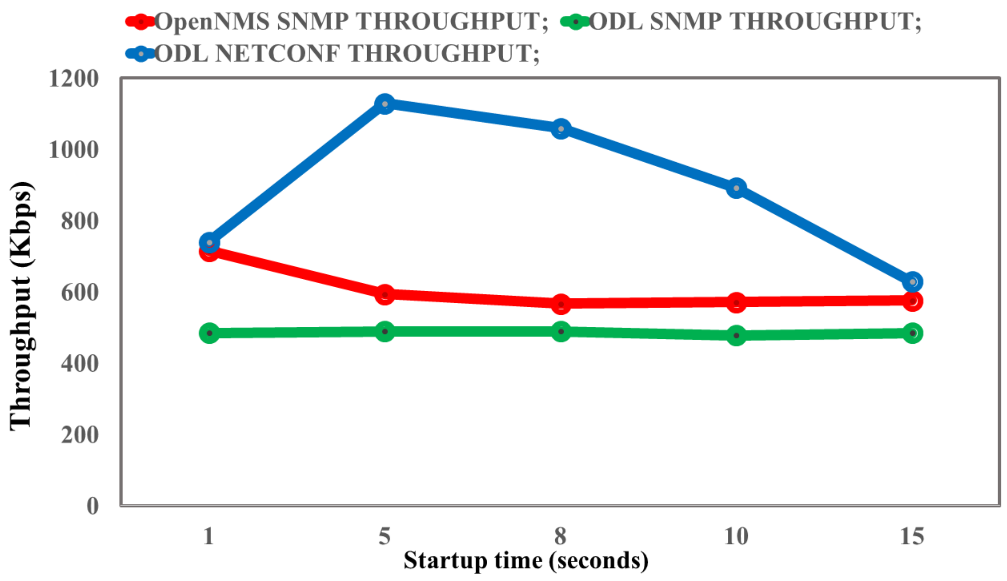


FIGURE 3.8: NETCONF and SNMP GET throughput (in Kbps) from 10,000 Users for ODL and OpenNMS.

presents a disadvantage regarding the throughput because it is a robust protocol where each frame has more bytes than an OpenNMS frame [142].

3.5 Summary

This chapter presented the results obtained from the performance evaluation study of IoT management platforms, using an IoT real environment, with a scenario of a smart lighting monitoring and management. The OpenDayLight platform has a modular design and supports IoT devices management and IoT network management. NETCONF and SNMP protocols were evaluated in the platform. The platforms (OpenDayLight and OpenNMS) were evaluated according to the packet size, latency, throughput, and average response time metrics.

The experiments results and corresponding comparison of the studied platforms concluded that OpenDayLight platform is the most suitable and robust for real environments. Its results were better than the other platforms and hosts both SNMP and NETCONF protocols natively on the system. The relevant protocols and platform will be used to create a new IoT management solution.

The OpenDayLight platform should be deployed and controls other real environment scenarios. Security features such as HTTPs/SSL for the IoT platform can also be considered in further developments.

Chapter 4

Proposal of an IoT Management Platform for Networks and Devices

In this chapter, the new IoT management platform, called M4DN.IoT, is proposed and described. The features and requirements that compose this platform are described and demonstrated through the UML diagrams. The architecture and demonstration of the M4DN.IoT platform is executed in a real lighting environment.

4.1 Software Requirements Engineering

Initially, it should conceptualize what constitutes requirements engineering: it is a discipline to develop a complete, consistent, and unambiguous specification that serves as the basis for an agreement among all parties involved, describing what the software product will do, not how it will be done. So, system requirements define what the system is required to do and under what circumstances they will be required to operate [143]. Requirements are categorized into two major groups: functional requirements and non-functional requirements.

- Functional requirements describe the functionalities or services of the system; have a localized effect because they affect only the part of the system where the functionalities are deployed.
- Non-functional requirements describe constraints or quality attributes related to the system; fix constraints on how functional requirements will be implemented and have system-wide effect as their satisfaction affects various components of the system.

Non-functional requirements are characterized by subjectivity, i.e., they can have different meanings for different people; relativity because their interpretation and importance depend on each system and their realization is relative and interactivity because they interact with each other, negatively or positively affecting each other. These requirements should to give objectivity and can be subdivided into the following requirements:

- Usability requirements, which are related to user interface, training material, system documentation, human factors;

- Reliability requirements, which are linked to failure frequency, recover-ability, predictability, availability of use, the degree of fault tolerance, system scalability;
- Performance requirements related to efficiency, processing response time, use of computational resources, throughput flow, accuracy, availability, resource use;
- Security requirements related to the privacy, integrity, and authenticity of system data;
- Distribution requirements, which relate to the distribution of the executable version of the system. Distribution requirements are critical for systems with a high volume of users;
- Requirements for ease of support, ease of adaptation and maintenance, internationalization, configurable;
- Implementation requirements, resource limitations, languages and tools, hardware, etc.;
- Interface requirements, restrictions imposed by interfaces with external systems;
- Operations requirements, which are related to the management of the system in the operating environment;
- Legal requirements, which are related to using licenses;
- Hardware and software restrictions, which are related to hardware and software to develop and run the system include the client platform, server platform, communication protocol, etc;
- Functional requirement is not a function, it is a functional need (a function) that the software must meet. A feature will be played by an actor.

Functional requirements are concerned to the functionality and services of the system, i.e., the functions the system must provide to the customer and how the system will behave in certain situations, through use-case diagrams, sequencing, and activities as may be seen in Section 4.2.1.

According to [144], a requirement is a statement about the desired product that specifies what it should do or how it should operate. When directed to software development, requirements should reflect the purpose of a system as well as the needs of its users, thus favoring the establishment of bases for the development of successful projects regarding to cost and quality [145].

4.1.1 Requirements Engineering Process

The process is organized into activities that become inputs and outputs. The description of the process is an important activity because it allows the knowledge to be reused in other situations [143], i.e., once someone has faced a problem and this solution has been documented, this can help other people who pass through the same kind of problem at another time.

Figure 4.1 can translate the process into a diagram of inputs and outputs that would be represented as shown in this figure [146].

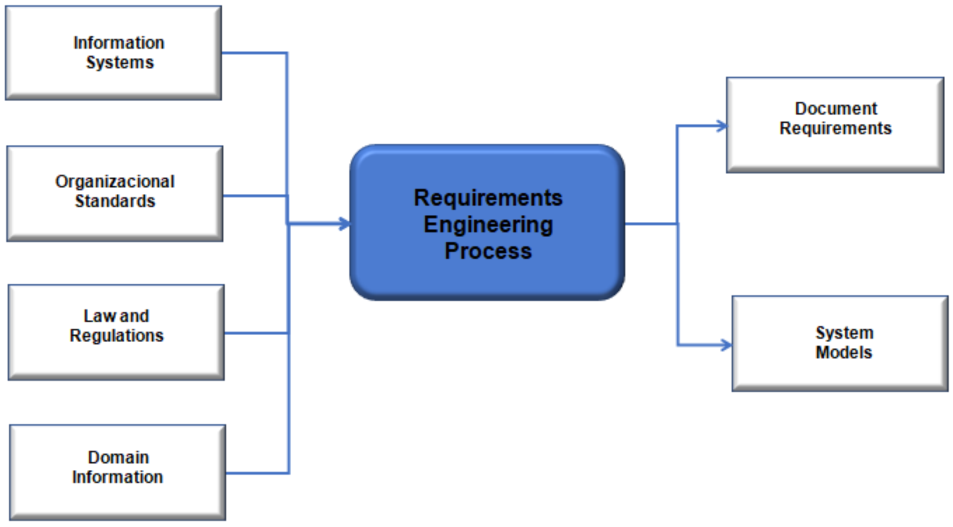


FIGURE 4.1: Requirements Engineering Process: input and output process.

The process inputs are the following:

- Information about existing systems: information about the system features that will be replaced or systems that iterate with the specified system;
- Organizational standards: standards that are used by the organization and that should be taken into account during the development of the application, quality management, methodology, etc.;
- Laws and regulations: legal issues that are external to the system should be observed so that rules are not violated;
- Domain information: general information about domain application.

On the outputs of the process there are the following:

- Document requirements: this will be a more detailed specification of the system as to what can be produced;
- System models: are diagrams that describes the system of different angles.

The new IoT management system utilizes a survey as an input process where it verifies all the existing systems. Regarding organizational standards and regulations, all care was used due to the application being accessed by an external or internal network (application domain) of the institution and all the requirements were duly documented.

The system modeling should be presented with Unified Modeling Language (UML) diagrams in Section 4.2.1 and detailed documented. In addition to this document, there was also the development of user guide and installation guide documentation.

4.1.2 Theoretical Fundamentals of Requirements

In the case of software applied to IoT, in some types of systems the ethnography becomes difficult for the analyst, given the specificity of the business and the difficulty of devices currently available [145]. Thus, two theoretical foundations were used to obtain the IoT management solution and specified below.

Reuse of requirements

Considering the good practices of software engineering, there was the reuse of the generated knowledge by performance evaluation of the IoT management protocols and platforms. The reuse of requirements saves time and effort since reused requirements have already been analyzed and validated in other systems. In this context, the OpenDayLight platform was reused as an application Back-end since this platform already supported most of the network and device management protocols effectively as evaluated in Chapter 3.

Prototyping

A system prototype is the initial version of the system that will be evaluated first during the development process. In software systems, prototypes are often used to assist in the elicitation and validation of system requirements. Another factor to be considered is the rapid development of prototypes, i.e., it is fundamental that soon available for the elicitation process. As benefits of prototyping, the following can be highlighted:

1. The prototype allows users to experiment and discover what they really need to support their work;
2. Establishes feasibility and usefulness before high development costs have been realized;
3. Essential for developing a user interface that meets the needs of system users;
4. Can be used for system testing and documentation development;
5. Forces a detailed study of requirements, revealing inconsistencies and omissions.

For the development of the new platform, it was used the prototyping method where through prototypes one can simulate the current operation of the system, and this will serve as a basis for the customer to feel much more comfortable to criticize, suggest changes, corrections, and the discovery of the need for requirements. The discovery of new requirements is crucial for the developing system since before that the system would be incomplete. This software to be developed must be constantly evolving and never be delivered to the client at the end of the process, i.e., at the beginning only some requirements will be developed and validated, but later the new prototype will have new interfaces so that the end user can whether it agrees or not and whether new requirements are not relevant.

4.1.3 Requirements analysis

In the requirements analysis step, there are a checklist that must be followed. The first check is if the project is immature, i.e., if the requirements include premature project or implementation information, then the combined requirements are observed. It is checked whether the description of the requirement describes a single requirement or whether it can be described in several different conditions. Then, a critical analysis is performed to come to the conclusion of which requirements are actually required and which are not. It should also be noted that the requirements do not imply the use of a non-standard hardware platform.

It should also be checked whether the requirements are consistent with the business objectives defined in the requirements document. If there are requirements that can lead to different interpretations, that is, if there are ambiguous requirements. The realism of requirements must be checked, that is, the requirement is realistic in relation to the technology used to implement the system and ultimately whether the requirement can be tested.

Functional Requirements

In the Table 4.1, the functional requirements to build the system are described.

TABLE 4.1: Functional Requirements of the IoT management solution (M4DN.IoT).

Use Case	Description
UC 1	User registration
UC 2	User authorities
UC 3	Administrator gateway registration
UC 4	Monitoring and control network information
UC 5	Monitoring and control device information
UC 6	Integrate solution with In.IoT Middleware

UC 1: User registration

The user story contemplates the user can register to gain access to the system. The login screen for users is also included in this user story.

UC 2: User authorities

In this user story, the user must have different privileges to the system. This permission is designed by the administrator user. Only the administrator authorities, view (read-only) should be created at first.

UC 3: Administrator gateway registration

The administrator user must be able to create, edit, and delete a management system gateway. IPv4 or IPv6 information must be assigned to each gateway for the automatic identification of devices through management protocols.

UC 4: Monitoring and control network information

The system must monitoring and managing the network according to the user's request for the system. Information such as: *i)* connection status, *ii)* IPv4 or IPv6 information, and *iii)* the supported network management protocols must be available.

UC 5: Monitoring and control device information

The system must monitoring and managing the devices according to the user's request for the system. Information on energy consumption and turn on/off device must be available at the first moment.

UC 6: Integrate solution with In.IoT Middleware

A modular solution must be built due to the integration with the In.IoT middleware system. The end customer must decide whether to use the middleware platform.

Non Functional Requirements

Usability is encompassed within quality and aims to ensure a part of the efficiency and effectiveness of the system. Efficiency refers to a productive interaction between the user and the system, allowing tasks to be performed with less effort under a pleasant interface. Thus, the solution must create an interface to enable the user to complete the task and achieve their goals in the system in a responsive and user-friendly.

The software must ensure data security as well as access permissions to its features, such as user password encryption (SSL, MD5, and ECC encryption) and free access to system menus according to the user hierarchy. When it comes to an IoT solution with confidential user information, this item becomes indispensable.

There is no point in having a secure, interactive, and reliable system if it consumes a lot of computer resources and takes time to perform the processing. A slow system is subject to criticism from users, even if it is functional. Software performance can be improved by using object-oriented programming techniques, memory management, threads, and code optimization. Other factors like the use of NoSQL banks due to the "Big Data" concept where different devices are constantly generating information in an IoT environment are preponderant issues to improve the performance of a solution.

4.1.4 Requirements Modeling

The M4DN.IoT is a network management platform and IoT devices capable for mapping and controlling the connected network and devices. The Unified Modeling Language (UML) notation should be used as a way to represent concepts of objects and relations of the application.

The system must perform network parameters assignment and query operations for devices through network protocols and devices such as NETCONF and CoAP. This interaction must be performed in a standardized way using REST API service.

The user can view the data in a responsive Web platform, i.e., it is automatically reweighted according to the equipment used. Platform access can be done on a local or remote network. The only requirement is to have access to the Internet and a browser. The smart lighting scenario was used to be managed and controlled through a WEB platform called M4DN.IoT. This application can be used in a residential environment by turning on/off according to people present in the room and lighting available.

The use case diagram presented on Figure 4.2, describes the management actions that the user can perform on the M4DN.IoT platform.

Precondition

1. The user must have logged and authorization from the system.

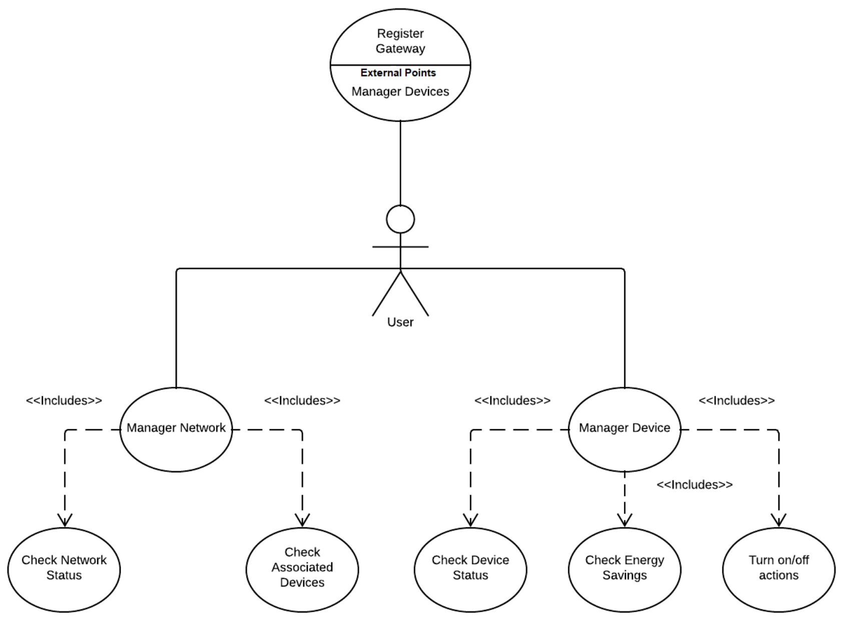


FIGURE 4.2: Use Case Diagram of the M4DN.IoT platform.

Primary Event Flow

1. When a user start the application, the IoT network management is initialized.
2. The user can request the state and quality of the connections.

Second Event Flow

1. When a user start the application, the IoT device management is initialized.
2. The user can request the turn on/off and state devices.

Post-condition

1. The network and connected devices must be managed.

The M4DN.IoT platform does not manage devices that do not have certain protocols developed, but the development was done in a modular way, i.e., one can create a plug-in with the proposed protocol for the management of others devices. The main problem is to create a modular platform with a friendly and responsive user interface in which any device connected to the network must be managed. The platform and device security requirements must be considered for development.

The modeling of the solution is demonstrated with the activity diagram in figure 4.3 and sequence diagram in figure 4.4.

Actors:

1. User and connected devices.

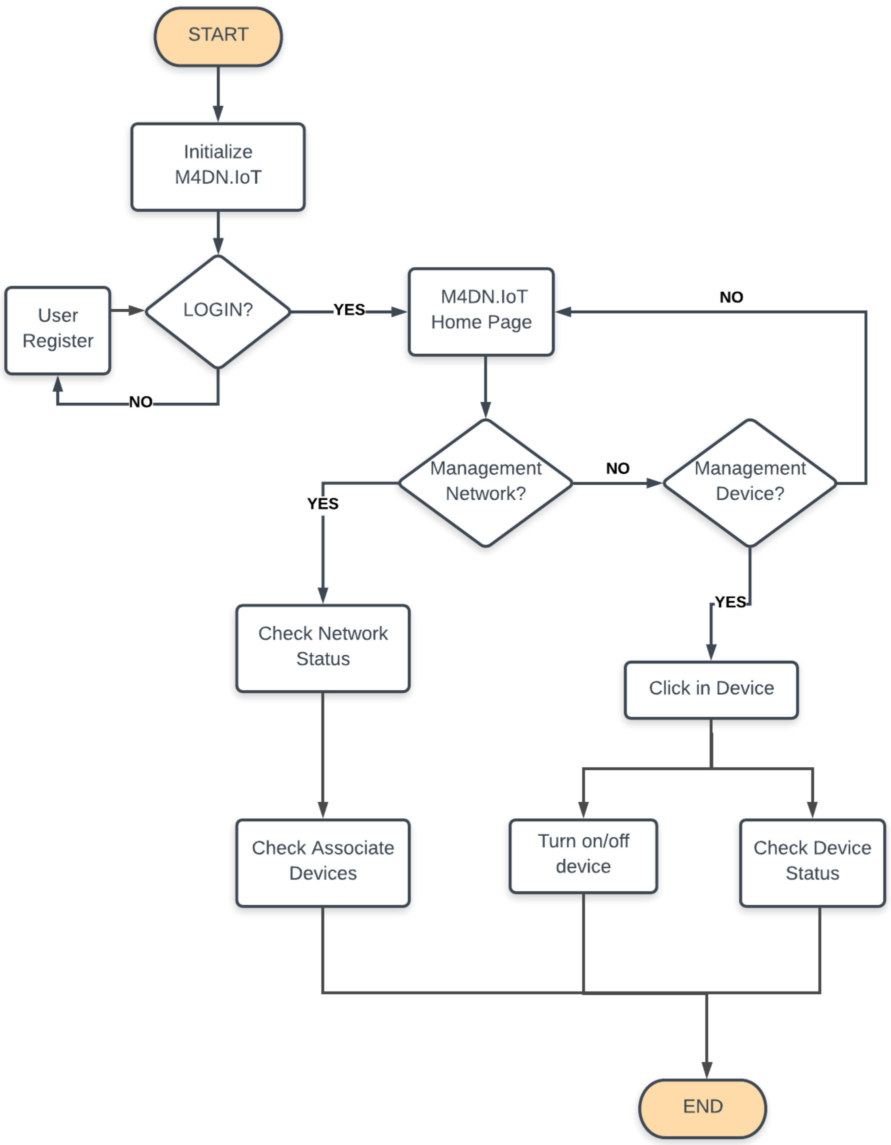


FIGURE 4.3: Activity Diagram of the M4DN.IoT platform.

Precondition:

1. The user must have logged and authorization from the system.

Basic Flow

1. The use case starts when the client accesses the system.
2. By clicking on the connections, the system shows information about the status and quality of the link.

3. When a user clicks on the gateway, the system displays information on the number of associated devices and the supported protocols.

4. When clicking on the device, the system shows device status information, amperage, voltage and IPv6 address. The user has the action of turn on/off the device.

5. The System sends a message of success or failure of the requested request and the use case ends

Post-condition:

1. The network and connected devices must be managed.

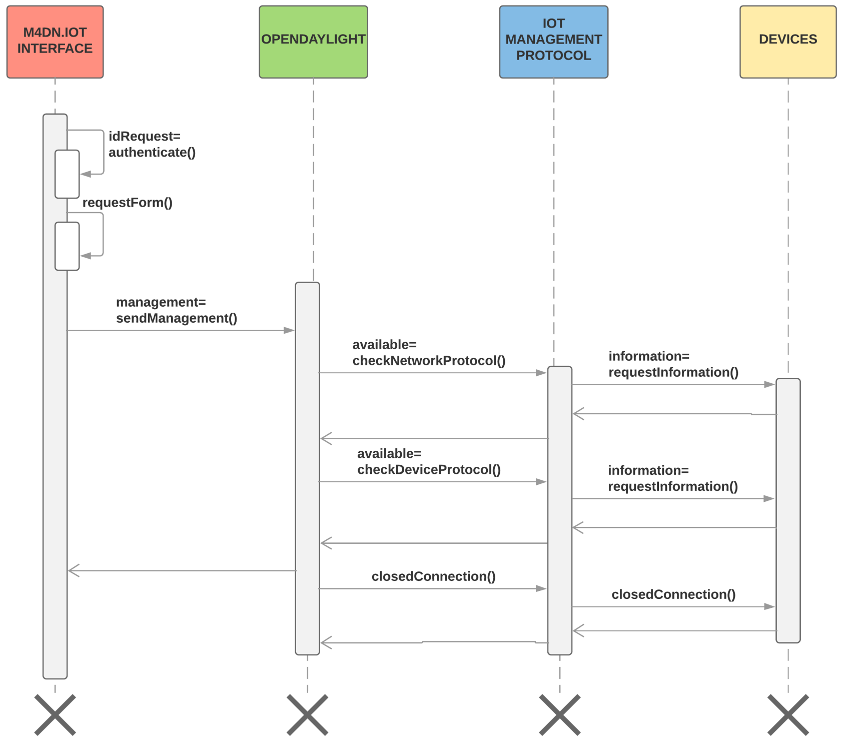

M4DN.IOT INTERFACE
OPENDAYLIGHT
IOT MANAGEMENT PROTOCOL
DEVICES
idRequest= authenticate()
requestForm()
management= sendManagement()
available= checkNetworkProtocol()
information= requestInformation()
available= checkDeviceProtocol()
information= requestInformation()
closedConnection()
closedConnection()

FIGURE 4.4: Sequence Diagram of the M4DN.IoT platform.

This platform is designed to work independently or in conjunction with the In.IoT [147] middleware platform as requested by customers.

4.1.5 System Requirements

As seen in Section 4.1, system requirements are a non-functional requirement that makes up the minimum and recommended hardware and software requirements described following..

Minimum Requirements

- A Linux Server Ubuntu 14.02 LTS, 64 bits;
- Any CPU (Intel i3 with 3.60GHz recommended);
- 4 GB RAM, 40 GB HDD Free Space;
- Any GPU that is compatible with OpenGL 3.2. (integrated graphics cards Intel HD 4000 or above).;
- Any Browser compatible with HTML 5 and CSS 3.0, e.g, Google Chrome Release 77;
- Java Development Kit (JDK) 7.0
- A tablet or smartphone with Android 4.0 Ice Cream Sandwich;

Recommended Requirements

- A Linux Server Ubuntu 14.04 or 14.06 LTS, 64 bits;
- Any CPU (Intel i5 with 3.60GHz recommended);
- 8 GB RAM, 80 GB HDD Free Space;
- Any GPU that is compatible with OpenGL 3.2. (integrated graphics cards Intel HD 4000 or above).;
- Any Browser compatible with HTML 5 and CSS 3.0, e.g, Google Chrome Release 77;
- Java Development Kit (JDK) 8.0;
- A tablet or smartphone with Android 4.4 Kit Kat.

Table 4.2 summarizes the minimum hardware and software requirements for installation and use of the M4DN.IoT platform.

TABLE 4.2: System Requirements minimum of Hardware and Software to installed the M4DN.IoT platform.

System Requirements	Platform Server	Mobile
CPU	Intel Core I3	Snapdragon 210 Quad-core
Clock Speed	3.60 GHz	1.0 GHz
RAM	4GB	4GB
ROM	40GB	16GB
Operational System	14.04 LTS	Android 4.0 Ice Cream Sandwich
Communication	WiFi and Ethernet	WiFi

4.2 Proposal of a new IoT Management Platform (M4DN.IoT)

The M4DN.IoT (Management for Devices and Networks in IoT) is a platform for managing the end devices (i.e. momote sensors, for example) and network elements (i.e. a gateway) that compose several IoT environments [148]. The scenarios of security control of a residence, the management of a lighting solution, demand side management [149], among others, can benefit from this application.The platform specifies a data and information model with the objective of standardizing the data format used in communication between applications, services, and devices. The status of the devices (on/off) and the id (device identification) are examples of features used in the information model for device management. On the other hand, the state and quality of the connections between network devices and the devices determine the information model for IoT network management. Also, for accessibility and integration with other systems, the platform makes use of industry-standard protocols and standards for data models, such as NETCONF, CoAP, and RESTfull API for performance evaluation in this IoT environment [120] [121] [122].

4.2.1 System Architecture

The M4DN.IoT platform was deployed in a prototype for the local or remote management subsystem [148]. The performance of the proposed solution was evaluated by employing them in the management of a public/residential lighting scenario. The components considered in the platform were the communication patterns (REST API, NETCONF/RESTCONF, CoAP, among others) through OpenDayLIght platform thus standardizing the data for consumption of other platforms.

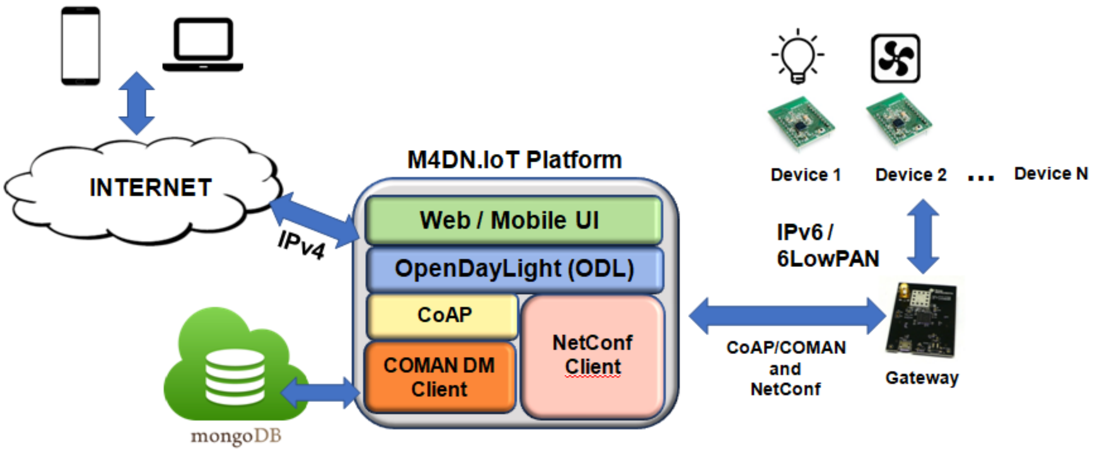


FIGURE 4.5: Architecture of the M4DN.IoT platform and its main components (used in a real environment scenario).

Figure 4.5 shows the layered architecture of the deployed solution. The application layer includes a responsive user interface using the bootstrap an AngularJS and NodeJS with the ODL platform. The database chosen was the MongoDB NoSQL database due to being a schemaless database, i.e., without fixed schema, that is an important feature to Big Data generated by connected devices. Both IoT management platforms provide the RESTful Web service to manage the equipment through the NETCONF and SNMP.

Analyzing the characteristics of the devices and the fact that many asynchronous actions, the software was developed employing a modular approach based on events.

4.2.2 Methods and models of software design

Software design comprises the design, specification, and prototyping of the "external" and "internal" parts of the software. The external part comprises the conceptual model of the application and the user interface. The inner part comprises the architecture of software components and the algorithms and data structures that implement these components. The diagrams will be used to represent data flow and related entity of the solution.

Data Flow Diagram

The data flow diagram (DFD) is a graphical representation of the data "flow" through an information system, modeling its process aspects. It provides only a view of the system, the structured view of the functions, i.e., the flow of data. Often, they are a preliminary step used to create an overview of the system that can be further elaborated. A DFD shows what kind of information will enter and exit the system, where data will request and where the data will be stored.

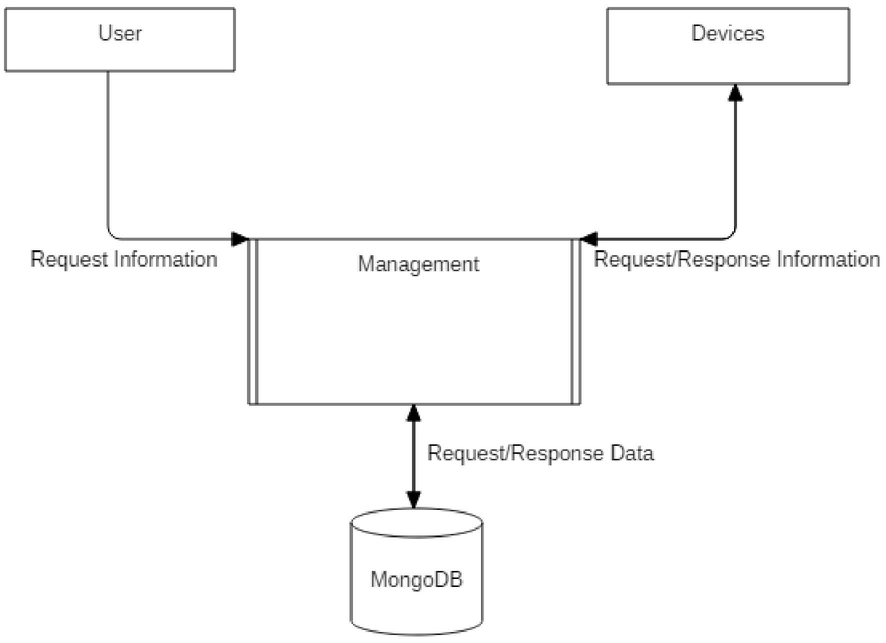


FIGURE 4.6: Data flow diagram of the M4DN.IoT platform.

The user requests a process of management or request information of devices this request should verify in the database if the information is the most current, otherwise,the process must request from the network or device that up-to-date information that can be seen by the data flow diagram, presented in Figure 4.6.

Relationship Entity Diagram

Relationship entity diagrams (REDs) are related to data structure diagrams (DSDs), which focus on the relationships of elements within entities rather than relationships between the entities themselves. RE diagrams are also often used in conjunction with data flow diagrams (DFDs), which map the flow of information to processes or systems.

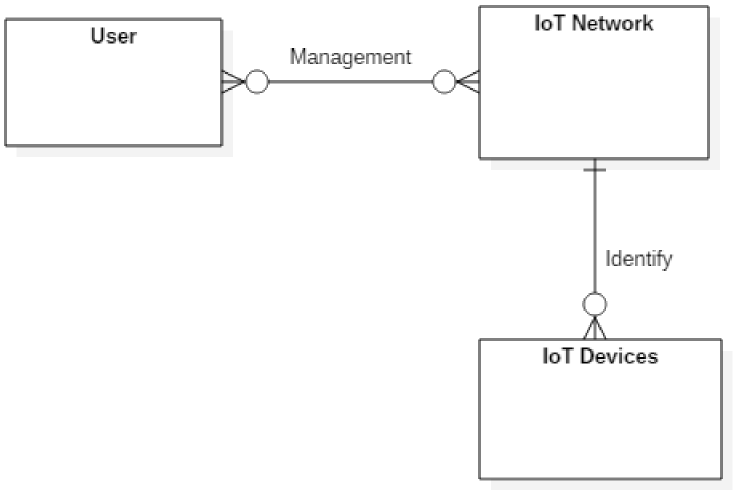


FIGURE 4.7: Relationship entity diagram of the M4DN.IoT platform.

Exemplifying the entity relationship diagram on Figure 4.7, the system user can manage several different IoT networks. Each IoT network is able to identify devices with context-aware. Summarizing, the user will be able to manage the IoT networks and devices.

4.2.3 Languages and Technologies Used

The important point is the technologies that were used for the development of the M4DN.IoT platform (where this application was developed), i.e., the possibility to use certain application modules according to the need of the IoT scenario that will be managed.

Table 4.3 summarizes the technologies used to the development of the platform.

Each technology used is summarized and its operation will be described as follows:

AngularJS is a technology in javascript, open source that is maintained by Google [150]. Its purpose is to increase applications that can be accessed by a Web browsers and is modeled by Model-View-View-Model (MVVM) in an effort to facilitate both application development and testing. Its use in the solution was decisive because most browsers support

TABLE 4.3: Technologies used in M4DN.IoT platform.

M4DN.IoT Characteristics	Technologies
Front end	AngularJS and PHP
Back End	NodeJS and REST API
Frameworks	Bootstrap and Sigmajs
Protocols	NETCONF, CoAP and COMAN DM
3PP Plataforms Supports	OpenDayLight
Java SDK	Java 7
NoSQL DaaS	MongoDB (mLAB)

both javascript and AngularJS. This requirement standardizes the solution for operation in any commercial browser.

PHP is an open source scripting language that is widely used, especially suitable for Web development and that can be embedded within HTML [151]. It is used as a tool to pre-process data that will be available at the Web page.

Node.js is a JavaScript code interpreter with open source focused on migrating client-side Javascript to servers [152]. As discussed earlier, scalability is one of the important requirements for IoT management. Its goal is to help programmers to create high-scalability applications (such as a Web server) with codes capable of handling tens of thousands of concurrent connections on a single physical machine.

A **RESTful API** is a method of allowing communication between a Web-based client and server that employs representational state transfer (REST) constraints [153]. This method is able to standardize the queries to the IoT management, being possible any application consume data of the same.

Bootstrap is the most popular HTML, CSS, and JavaScript framework for developing responsive, mobile-first Websites [154]. This framework was used to create one application with support in websites and mobile application. Thus, the system user can access the application of any device connected to the internet.

Sigma is a JavaScript library dedicated to graph drawing [155]. It makes easy to publish networks on Web pages and allows developers to integrate network exploration in rich Web applications. This framework was used to create the network topology of the IoT management solution.

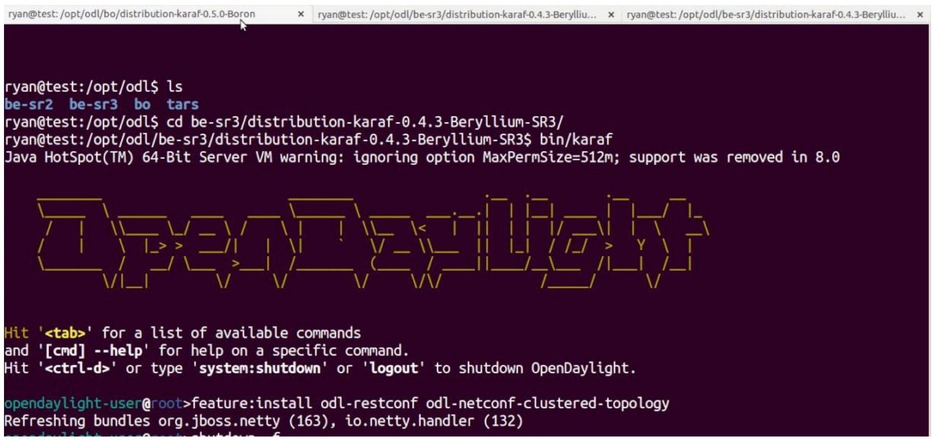
MLab is a fully managed cloud (DaaS) database service that hosts **MongoDB databases** [156]. MLab runs on cloud providers Amazon, Google, and Microsoft Azure, and has partnered with platform providers as a service. This database was chosen due to the NoSQL database schema-free and because it is hosted as a cloud service, thus guaranteeing high scalability.

4.3 Solution Demonstration and Validation

This section focuses in the demonstration and validation of a new IoT Management platform to validate the use of the new platform in a real IoT environment, the M4DN.IoT.

4.3.1 Front-end and Back-end usability

OpenDayLight is the platform with the most support for IoT management protocols. When there is a need to use another protocol, it can be developed in feature forms on the platform, shown in Figure 4.8, thus ensuring modularity.



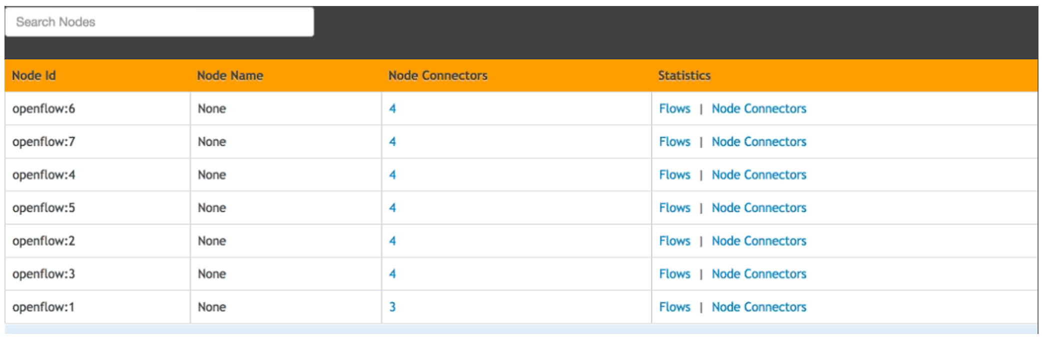
ryan@test: /opt/odl/bo/distribution-karaf-0.5.0-Boron | ryan@test: /opt/odl/be-sr3/distribution-karaf-0.4.3-Berylliu... | ryan@test: /opt/odl/be-sr3/distribution-karaf-0.4.3-Berylliu...

```
ryan@test:/opt/odl$ ls
be-sr2  be-sr3  bo  tars
ryan@test:/opt/odl$ cd be-sr3/distribution-karaf-0.4.3-Beryllium-SR3/
ryan@test:/opt/odl/be-sr3/distribution-karaf-0.4.3-Beryllium-SR3$ bin/karaf
Java HotSpot(TM) 64-Bit Server VM warning: ignoring option MaxPermSize=512m; support was removed in 8.0

Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'system:shutdown' or 'logout' to shutdown OpenDaylight.

opendaylight-user@root>feature:install odl-restconf odl-netconf-clustered-topology
Refreshing bundles org.jboss.netty (163), io.netty.handler (132)
```

FIGURE 4.8: Illustration of the installation of the protocols.

The OpenDayLight was chosen as the back-end of the new platform. The evaluation of OpenDayLight and its protocols were described in Chapter 3. Front-end requirements are a challenge for OpenDayLight because the interface is not user-friendly and it is designed for users with knowledge in development technologies. Security requirements such as HTTPS are also issues that were not developed and are mandatory for an IoT management platform.

In Figure 4.9, each device must be added manually by the network manager. Through the OpenDayLight UI (DLUX), the management must be performed by Yang UI, where the user must be aware of REST API structure to manage the network devices, as can be seen in Figure 4.10



Search Nodes

Node Id	Node Name	Node Connectors	Statistics
openflow:6	None	4	Flows | Node Connectors
openflow:7	None	4	Flows | Node Connectors
openflow:4	None	4	Flows | Node Connectors
openflow:5	None	4	Flows | Node Connectors
openflow:2	None	4	Flows | Node Connectors
openflow:3	None	4	Flows | Node Connectors
openflow:1	None	3	Flows | Node Connectors

FIGURE 4.9: Illustration of the installation of the protocols.

The new platform M4DN.IoT was developed to solve the issues discussed above. Interface requirements have been solved through a friendly and responsive platform, that is, can be used in any device connected to the Internet.

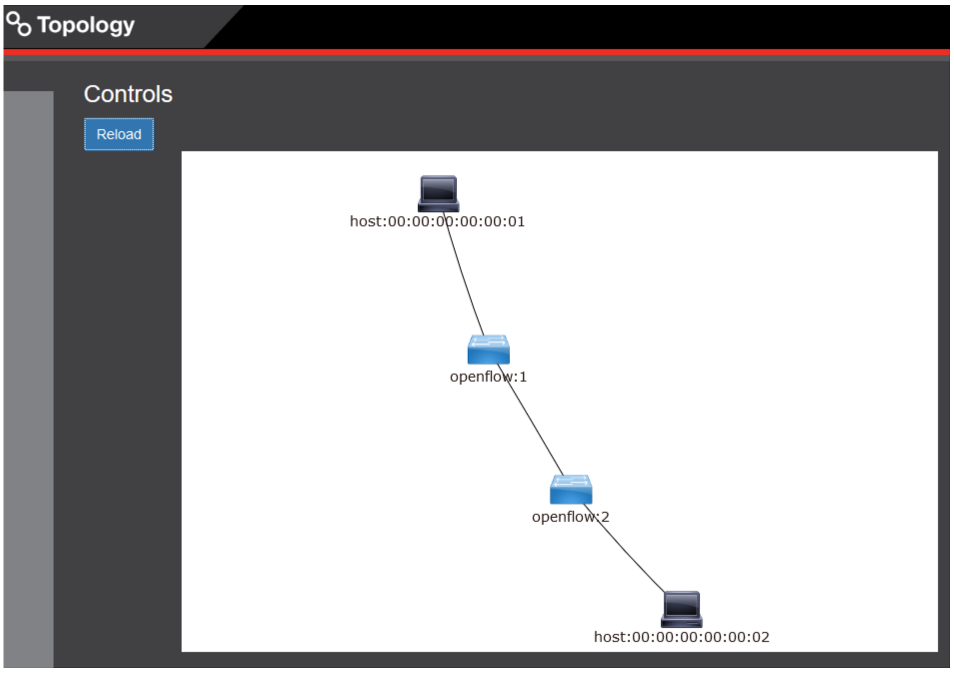


FIGURE 4.10: Illustration of the nodes register in OpenDayLight UI (DLUX).

Security issues under development are: *(i)* use of the HTTPS protocol and *(ii)* develop the ECC protocol and DTLS over JWT for protocol communication of the network with devices.

In the next section the operation of each M4DN.IoT functionality will be described.

4.3.2 Main Functionalities and Operation

The platform contains a user registration feature, Figure 4.11 allows user to perform a query and the registration of new users with the following information: Username, Password, and Email address, where all the information is mandatory. This feature could be used or not in the platform, due to the control user access is provided of the In.IoT middleware when integrated [147].

The prototype performs communication between components via management protocols. Thus, the sensors generate information as new entries in the database, and the platform can perform data queries to the inserted data. For example, when the light sensor reads a new value, the platform inserts that value into the database. The application, in turn, can read this value and use it to set its operating parameters. The User must register the Gateways that will be managed by the platform. When they are registered, the M4DN.IoT automatically finds all the devices interconnected through requests of the NETCONF or SNMP protocols thought the OpenDayLight platform as a bridge.

The goal of this modular and dynamic platform is to make the centralizing application fully capable of managing other devices, without making changes to the application. The

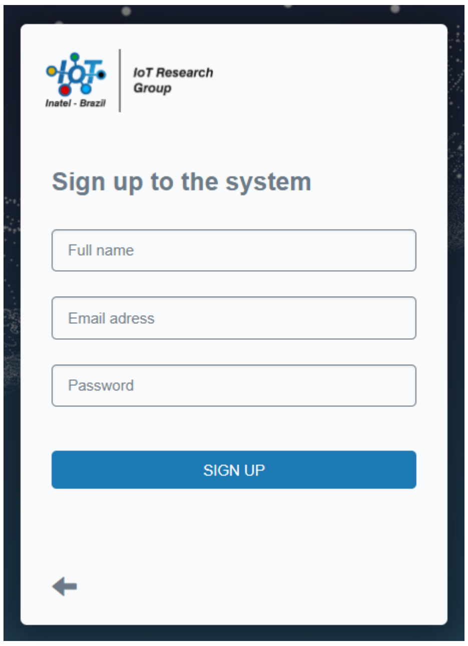
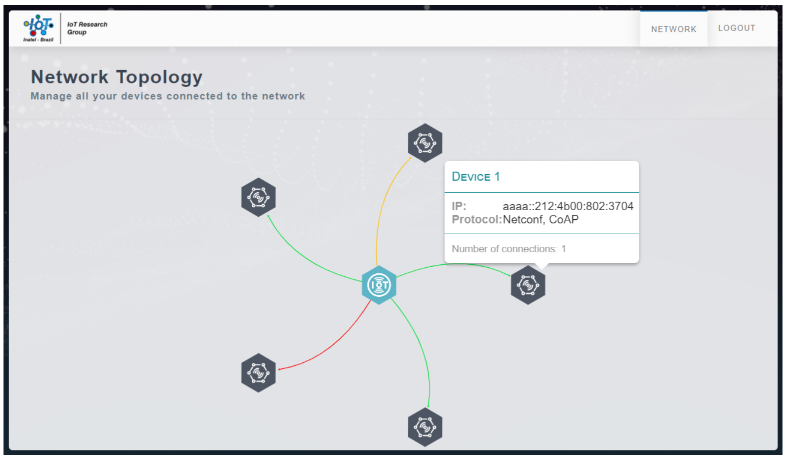


FIGURE 4.11: User Register at the M4DN.IoT Platform.

M4DN.IoT platform obtains information of the IPv6 gateway and devices, protocols supported, in addition to other information. Another important requirement, shown in Figure 4.12 is the connection management between the gateway and the Momote, which are represented in the following colors: *i)* Green: successful connection status, *ii)* Yellow: status connection warning due to response delay of device, and *iii)* Red: Connection status failed.

M4DN.IoT collects and stores the network information for the managed devices, such as Device Name, Valid IPv6 Address, Number of connected devices, and Protocols supported according to Figure 4.12.



Device 1

IP: aaaa::212:4b00:802:3704
Protocol:Netconf, CoAP

Number of connections: 1

FIGURE 4.12: Information from the IoT Network Management at the M4DN.IoT Platform.

Also, the platform can manage devices located by the Gateway. The application can turn on/off a particular light or fan device. Other relevant information that can be obtained of the sensors are: *i)* voltage, *ii)* amperage, and *iii)* the status that the device is on or off.

This information can be observed by the Figure 4.13.

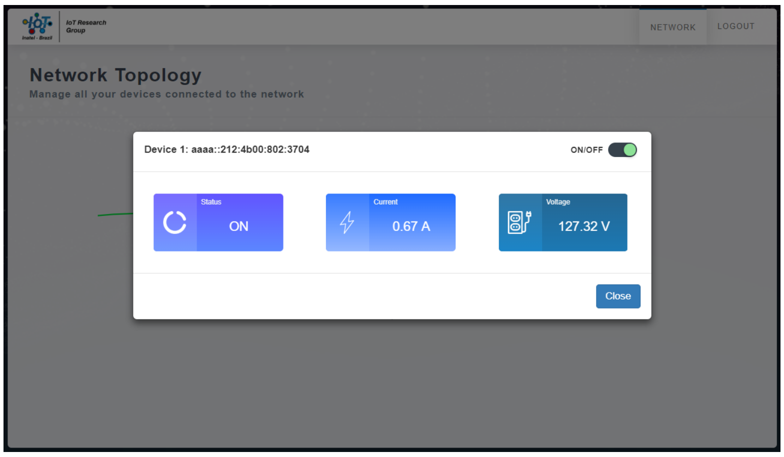


FIGURE 4.13: Information from the IoT Device Management showing sensors data at the M4DN.IoT Platform.

In Figure 4.14, it was observed that if we considered a priory the worst scenario with start-up in 0 seconds, M4DN.IoT obtained about 200 errors in user requests compared to ODL. This result is due to the amount of layer security and transport used, where in M4DN.IoT was developed the HTTPs and JSON Web Tokens (JWT) for Rest-Full API. Another negative point of ODL is that it does not have the HTTPs protocol developed so it is possible to obtain information with an external sniffer.

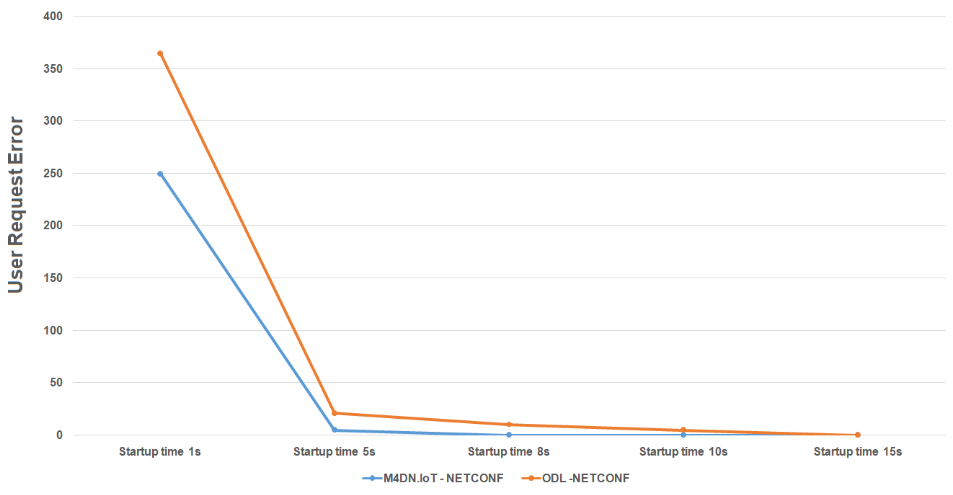


FIGURE 4.14: User Request Error of NETCONF request with 10,000 Users for M4DN.IoT and OpenDayLight (ODL).

Based on this survey [17], it is concluded that security, context-aware, and the standard model of messages still in an early stage. These requirements have not been completely explored. Interoperability and heterogeneity are requirements that have been maintained from the ODL system and are therefore achievable at M4DN.IoT. Thus, M4DN.IoT was developed to resolved these requirements. In terms of the security, the HTTPs and JWT was development to guarantee the frontend and backend security respectively. By using these security technologies, it was eliminated a large part of the security header that the OpenDayLight project used. Thus, as seen in 4.14, the number of requests with errors were smaller in M4DN.IoT compared to OpenDayLight. Another resolved point was regarding context-aware, where the M4DN.IoT system performs the discovery and management of the gateway and connected devices automatically. All such communication and information use Rest API as a standard model of messages.

Currently, the platform is deployed together with the In.IoT middleware platform [147] in a modular way, that is, it can work together or not with the client's requesting middleware.

4.3.3 Future Developments

Finally, the relevant requirements for future developments in the M4DN.IoT application are described as follows.

- Security: Use HTTPS, ECC, DTLS, or other security protocols for the platform to make reliable;
- The functional and non functional requirements are known issues to future development;
- Perform self-acknowledgment and registration of the Gateway in the network;
- Include the platform in other scenarios of real environments;
- CoMI protocol development in M4DN.IoT;
- Integrate the Blockchain resources to resolve security and communication problems;

4.4 Summary

This chapter described the most relevant features and requirements that composed the new IoT management solution, called M4DN.IoT. M4DN.IoT is a modular platform that manages the state and quality of the connections between the network devices that determine an IoT network management.

The platform specifies a data and information model with the objective of standardizing the data format used in communication between applications, services, and devices. The final result has executed this platform in a real lighting environment.

Chapter 5

Conclusion and Future Works

To conclude the dissertation, this chapter addresses the lessons learned along this study, summarizes the main conclusions, and suggests topics for further research works.

5.1 Lessons Learned

This dissertation analyzed the network and devices management for IoT from different perspectives. The lessons learned from this study are summarized hereafter. First, from the architectural point of view, IoT network management systems is well adopted by IoT reference architecture and research attempts [4], [9]. However, due to the distinct network topologies, heterogeneity between connected devices and protocols without a common standard, there are possible studies to perform in this context. Scalability and interoperability have importance in IoT applications and some of the studied technologies present solutions for them [157], [98], [158], [159]. Then, larger studies of recent researches were analyzed to discover the main challenges and open issues in IoT. Security and interoperability are top priorities for IoT network management applications, followed by performance, reliability, and scalability. So, there are some fundamental features that a network management should provide and they are identified as follows: *i)* interoperability between the various devices and platforms available in real environments; *ii)* dynamic and adaptation security keep data integrity to guarantee the availability and QoS during execution; *iii)* context-awareness so that information of regarding the location and state of network objects is used to perform actions; and *iv)* scalability to accept expansion and to operate correctly even in situations of intense use.

Another important aspect comes from the fact that there are many IoT management protocols and platforms where each one utilizes different data and standards format. An IoT environment has customer's devices with different data models. This is one challenge to create a new IoT management platform capable of managing customer's devices. Energy savings for restricted devices should not be the only relevant features. Network interoperability with heterogeneity devices is an important issue for IoT management platforms. Security is a key issue, due user's information trafficking on the network and can be easily breached.

Initially, the construction of programming code for the network management protocols was done to meet only the basic management characteristics. However, during the experiments, it was necessary to adapt the code so that there was a standard data model of communication and devices management, and this used a REST API technology. It was also necessary to assimilate how to source code compilation and every Momote configuration to simulate the real lighting environment in a laboratory environment for the development of management technologies.

Another important technology that helped in the development of protocols was Cooja [160]. Cooja is an emulator of the Contiki operating system to simulate an IoT network. Thus, each new implementation was simulated and debugged through Cooja to verify this new functionality running on the network.

5.2 Concluding Remarks

A large number of IoT devices demands management and control solutions for various services. Moreover, the exponential number of connected devices and their inherent constraints motivate the need for efficient management of IoT networks. Therefore, platforms that integrate these services are necessary. However, current IoT management platforms only partially attend the literature requirements. Overall, this dissertation presented the concept in detail, its enabling technologies, protocols, platforms, and the recent research addressing IoT management for networks and devices with the goal to create a new IoT management platform. Among the IoT features, solutions (protocols and platforms) that perform better in terms of scalability, interoperability, security, energy saving, etc. were studied.

Concerning network management, IoT network solutions continue based on the SNMP protocol. The support for each platform search for improving the latency, scalability, and robustness. NETCONF protocol was developed to be the natural successor of SNMP, as SNMP is focused on monitoring and not on network configuration. The OpenDayLight platform can be considered the best solution based on the supported protocols. For IoT devices management, ONEM2M open source approach and Xively proprietary technology were the evaluation with other technologies. It was observed that ONEM2M and Xively predict scalability and promote the integration of devices with local/remote management features remembering always the guarantee the heterogeneity and security.

This dissertation presented an overview of the protocols and approaches used for IoT devices and networks management where their motivation and technical challenges were identified. A comparative analysis of the studied approaches to choose the best technologies used to a new IoT network management platform was presented. The NETCONF protocol and OpenDayLight platform were used as back-end technologies. This solution provides requirements for automatic IoT network management and a user-friendly interface that provide information about the network devices (i.e. IP Address, connection status, and protocols supported) and connected devices (i.e voltage, amperage, and on/off status). This platform can be used in any equipment (desktop computer, smartphone, and tablet) and its access is available in any location.

5.3 Future Works

To conclude the study, some relevant future works on IoT management are identified and are presented as follows.

- Security service provisioning by using SDN network management;
- Business applications in real-time (with edge computing and fog computing using artificial intelligence contributions);
- Reducing network resources (hardware and bandwidth) and energy savings;
- Compatibility and Longevity: lack of standardized in M2M protocols and diversities in firmware;
- Intelligent Analysis and Actions: extracting insights from data for analysis;
- Privacy: data privacy and tracking devices;
- Regulatory standards for data markets are missing especially for data brokers and gateways;
- Include the platform in other scenarios of real environments, e.g., SDN network management.

References

[1] O. Bonaventure. “Computer Networking: Principles, Protocols and Practice”. In: *Saylor Foundation, Belgium, May 30, 2014*. IEEE. 2013, pp. 1–138.

[2] M Zorzi A. Zanella; N. Bui; A. Castellani; L. Vangelista. “Internet of Things for Smart Cities”. In: *IEEE Internet of Things Journal* 1.2 (2014), pp. 22–30.

[3] X. Du; Y. Xiao; M. Guizani & H. H. Chen. “An Efficient Key Management Scheme for Heterogeneous Sensor Networks”. In: *International Journal of Computer Science and Information Technologies* 2.1 (2011), pp. 2343–2347.

[4] T. Qiu; N. Chen; K. Lib; D. Qiaoc & Z. Fud. “Heterogeneous ad hoc networks Architectures, advances and challenges”. In: *Computer Network* 55.2 (2017), pp. 143–152.

[5] M. Slabicki & K. Grochla. “Performance evaluation of CoAP, SNMP and NETCONF protocols in fog computing architecture”. In: *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium* (2016), pp. 143–152.

[6] Varun M Tayur & Dr. R. Suchithra. “Internet of Things Architectures: Modeling and Implementation Challenges”. In: *Computational Systems for Health & Sustainability* (2015), pp. 9–13.

[7] N. Benamar; A. Jara; L. Ladid & D. E. Ouadghiri. “Challenges of the Internet of Things: IPv6 and Network Management”. In: *Eighth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing* (2014), pp. 30–53.

[8] L. Atzori; A. Iera & G. Morabito. “The Internet of Things: A survey”. In: *Computer Networks* 54.15 (2010), pp. 2787–2805.

[9] M. Meng; W. Ping & C. Chao-Hsien. “Data Management for Internet of Things: Challenges, Approaches and Opportunities”. In: *IEEE and Internet of Things Conference (iThings/CPSCom)* (2003), pp. 1144–1151.

[10] A. Passemard. “The Internet of Things Protocol stack - from sensors to businesss value”. In: *Internet of Things (IoT) talks* (2014), pp. 11–15.

[11] J. Ruiz; B. Linnyer; M. S. J. Nogueira; M. S. José & A. A. Loureiro. “MANNA: A management architecture for wireless sensor networks”. In: *IEEE Communications Magazine* (2003), pp. 116–125.

[12] P. F. Pires; E. Cavalcante; T. Barros; F. C. Delicato; T. Batista & B. Costa. “A platform for integrating physical devices in the Internet of Things”. In: *12th IEEE International Conference on Embedded and Ubiquitous Computing* (2014), pp. 234–241.

[13] C. Perera; A. Zaslavsky; P. Christen & D. Georgakopoulos. “Context Aware Computing for The Internet of Things: A Survey”. In: *IEEE Communications Surveys & Tutorials* (2013).

[14] D. C. Y. Vargas & C. E. P. Salvador. “Smart IoT Gateway For Heterogeneous Devices Interoperability”. In: *IEEE Latin America Transactions* (2016).

[15] P. Sethi & S. R. Sarangi. “Internet of Things: Architectures, Protocols, and Applications”. In: *Journal of Electrical and Computer Engineering* (2016).

[16] J. C. Silva; J. J. P. C. Rodrigues & M. L. Porença Jr. “IoT Network Management: Content and Analysis”. In: *Simpósio Brasileiro de Telecomunicações e Processamento de Sinais* (2017).

[17] Jonathan de C. Silva; Joel J. P. C. Rodrigues; Jalal Al-Muhtadi; Ricardo A. L. Rabêlo; Vasco Furtado. “Management Platforms and Protocols for Internet of Things: A Survey”. In: *IEEE Internet of Things Journal, February 15, 2019*. Vol. 19. Sensors. 2016.

[18] L. L. de Souza; P. H. M. Pereira; J. C. Silva; C. N. M. Marins; G. A. B. Marcondes & J. J. P. C. Rodrigues. “IoT Network Management Protocols: Practical Model and Evaluation”. In: *IEEE International Workshop on Communication, Computing, and Networking in Cyber Physical Systems (CCNCPS 2018)* (2018).

[19] J. C. Silva; P. H. M. Pereira; L. L. de Souza; C. N. M. Marins; G. A. B. Marcondes & J. J. P. C. Rodrigues. “Performance Evaluation of IoT Network Management Platforms”. In: *7th International Conference on Advances in Computing, Communications and Informatics (ICACCI 2018)* (2018).

[20] J. C. Silva; J. J. P. C. Rodrigues; K. Saleem; S. A. Kozlov; R. A. L. Rabêlos. *M4DN.IoT - A Network and Protocols Management Platform for IoT.*

[21] J. C. Silva; F. B. Brito; J. J. P. C. Rodrigues & G. A. B. Marcondes. “Estudos sobre Plataformas de Gerenciamento de Redes e Dispositivos para Internet das Coisas”. In: ().

[22] A. Clemm. 2007.

[23] A. P. Athreya & P. Tague. “Network Self-Organization in the Internet of Things”. In: *IEEE Communications Society Conference Sensor, Mesh and Ad Hoc Communications and Networks (SECON)* (2013), pp. 398–431.

[24] S. Lee; K. Levanti & H. S. Kim. “Network monitoring: Present and future”. In: *Computer Networks* (2014), pp. 84–98.

[25] J. F. Kurose & K. W. Ross. 2010.

[26] M. Gabdurahmanov & S. Trygg. “Analysis and Evaluation of Network Management Solutions”. In: *Degree Project in Computer ENgineering, Sweden 2016* (2016), pp. 1–84.

[27] O. Vermesan & P. Friess. “Internet of Things – From Research and Innovation to Market Deployment”. In: *River Publishers Series in Communication* 4.5 (2014), pp. 2787–2805.

[28] L. Sanchez; K. McCloghrie & J. Saperia. *Evolution of LTE in Release 13*. 3GPP, Nov. 2017. URL: `https://tools.ietf.org/html/rfc3139` (visited on 06/05/2018).

[29] L. L. Peterson & B. S. Davie. *Computer networks: a systems approach*. 5. Elsevier, 2012, pp. 1340–2200.

[30] C. Zhou & X. Zhang. “Toward the Internet of Things Application and Management: A Practical Approach”. In: *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks* 1.2 (2014), pp. 1–6.

[31] F. C. Delicato; P. F. Pires & T. Batista. “Middleware solutions for the Internet of Things”. In: *Briefs in Computer Science* (2013), pp. 10–21.

[32] C. Perera; S. Member; A. Zaslavsky & P. Christen. In: *Context Aware Computing for The Internet of Things: A Survey*. IEEE, 2013, pp. 414–454.

[33] B. Song; Y. Cheong; T. Lee & J. Jeong. “Design and Security Analysis of Improved Identity Management Protocol for 5G/IoT Networks”. In: *World Conference on Information Systems and Technologies* (2017), pp. 10–21.

[34] European Commission. *Deliverable D1.3 – Internet-of-Things Architecture IoT-A*. Tech. rep. European Commission, 2017, pp. 1–24. URL: `https://www.cs.mun.ca/courses/cs6910/IoT-A-D7_2.pdf`.

[35] Y. Ma; J. Chen; Y. Huang & M. Lee. *An Efficient Management System for Wireless Sensor Networks*. Sensors, 2010.

[36] J. Case; M. Fedor; M. Schoffstall & J. Davin. *RFC 1157 - Simple Network Management Protocol (SNMP)*. 3GPP, Nov. 2017. URL: `https://tools.ietf.org/html/rfc1157/` (visited on 02/05/2018).

[37] SNMP Research. *Simple Network Management Protocol*. SNMP, Nov. 2017. URL: `http://www.snmp.com/protocol/` (visited on 02/15/2018).

[38] Dra. E. S. SPECIALSKI. *Gerência de Redes de Computadores e de Telecomunicações*. Universidade de Santa Catarina, 2002.

[39] A. B. Ericsson. *Simple Network Management Protocol 5.2.5 (SNMP)*. Erlang, 2017.

[40] ASN.1 PROJECT. *Introduction to ASN.1*. ITU-T, Oct. 2017. URL: `http://www.itu.int/en/ITU-T/asn1/Pages/introduction.aspx` (visited on 02/15/2018).

[41] The OpenNMS Group. *OpenNMS - Open Networks Management System*. ITU-T, Apr. 2017. URL: `https://www.opennms.org/en` (visited on 02/15/2018).

[42] R. Enns; M. Bjorklund; J. Schoenwaelder & A. Bierman. *RFC 6241 - Network Configuration Protocol (NETCONF)*. IETF, Mar. 2017. URL: `https://tools.ietf.org/html/rfc6241` (visited on 02/15/2018).

[43] M. Bjorklund. *RFC 6020 - YANG - A data modeling language for NETCONF*. IETF, Oct. 2017. URL: `https://tools.ietf.org/html/rfc6020` (visited on 02/25/2018).

[44] H. Xu; C. Wang; W. Liu & H. Chen. “NETCONF-based Integrated Management for Internet of Things using RESTful Web Services”. In: *International Journal of Future Generation Communication and Network* (2012).

[45] C. Chappell. “White Paper Creating the Programmable Network: The Business Case for NETCONF/YANG in Network Devices”. In: *Heavy Reading* (2013).

[46] B. Pfaff & B. Davie. *RFC 7047 - The Open vSwitch Database Management Protocol*. IETF, Dec. 2017. URL: `https://tools.ietf.org/html/rfc7047` (visited on 02/13/2018).

[47] B. Davie; T. Koponen; J. Pettit; M. Casado; N. Gude & T. Petty. “Public Review for A Database Approach to SDN Control Plane Design”. In: *ACM SIGCOMM Computer Communication Review* (2017).

[48] R. Beryllium. *OpenDaylight Documentation*. Tech. rep. OpenDaylight Project, 2017, pp. 1–24. URL: `https://docs.opendaylight.org/`.

[49] C. Pastrone; M. Boella; M. Spirito; R. Tomasi & F. Rizzo. “A Jabber-Based Management Framework for Heterogeneous Sensor Network Applications”. In: *ACM SIGCOMM Computer Communication Review* (2008).

[50] M. Scheck. *Performance tests XMPP*. IoT Messaging Protocols, May 2017. URL: `https://iotprotocols.wordpress.com/2015/03/31/performance-tests-xmpp/` (visited on 01/03/2018).

[51] A. Stanik & O. Kao. “A proposal for REST with XMPP as base protocol for intercloud communication”. In: *7th International Conference on Information, Intelligence, Systems & Applications (IISA)* (2016), pp. 10–34.

[52] EU-Brazil research and development Cooperation. *D3.4 Network Management*. Tech. rep. IMPReSS Consortium, 2015, pp. 1–16. URL: `http://impressproject.eu/downloads/deliverables/D3.4_Network_Management.pdf`.

[53] H. Yan. “Smart Devices Collaboration for Energy Saving in Home Networks”. In: *Préparée à l’unité de recherche IRISA (UMR 6074)* (2014).

[54] H. Yan & F. Fontaine. “Patent 1454940: An adaptive proxy for compliance of the equipments to ieee 1905 network”. In: *Institut de Recherche en Informatique et Systèmes Aléatoires* (2014).

[55] IEEE. “IEEE Standard for a Convergent Digital Home Network for Heterogeneous Technologies”. In: *IEEE* (2013).

[56] S. Palm. “Connected home - Focus on Networked Power Save and Management”. In: *ACEEE Intelligent Efficiency Conference: Program* (2014).

[57] P. A. C. Neves & J. J. P. C. Rodrigues. “Internet Protocol over Wireless Sensor Networks, from Myth to Reality”. In: *Journal of Communications* 5.3 (2010), pp. 189–196.

[58] A. Sehgal; V. Perelman; S. Kuryla & J. Schönwälder. “Management of Resource Constrained Devices in the Internet of Things”. In: *IEEE Communications Magazine* (2012).

[59] G. Ayala; P. Poskal & E. Gamess. “SNMP JManager: An Open Source Didactic Application for Teaching and Learning SNMP v1/2c/3 with Support for IPv4 and IPv6”. In: *Seventh LACCEI Latin American and Caribbean Conference for Engineering and Technology (LACCEI’2009)* (2009).

[60] H. Mukhtar; K. Kang-myo; S. A. Chaudhry; A. H. Akbar; K. Ki-hyung & S. Yoo. “LNMP - Management Architecture for IPv6 based low- power Wireless Personal Area Networks (6LoWPAN)”. In: *Network Operations and Management Symposium* (2008).

[61] N. Kushalnagar; G. Montenegro & C. Schumacher. *RFC 4919 - 6LoWPAN: Overview, Assumptions, Problem Statement and Goals*. IETF, July 2017. URL: `https://tools.ietf.org/html/rfc4919` (visited on 01/03/2018).

[62] K. Kim; S. Yoo; D. Park; J. Lee & G. Mulligan. *Hierarchical Routing over 6LoWPAN (HiLow)*. IETF, July 2007. URL: `https://tools.ietf.org/html/draft-daniel-6lowpan-hilow-hierarchical-routing-01` (visited on 01/03/2018).

[63] EU-Brazil research and development Cooperation. *D7.3.2 Final Design and Implementation of the Configuration and Composition Manager*. Tech. rep. IMPReSS Consortium, 2015, pp. 1–16. URL: `http://www.compose-project.eu/sites/default/files/publications/D7.3.2%5C%20-%5C%20Use%5C%20cases%5C%20implementation%5C%20%5C%E2%5C%80%5C%93%5C%20Final%5C%20version.pdf`.

[64] The OpenNMS Group. *OpenNMS - Open Network Management System Wiki*. Network Working Group, Nov. 2017. URL: `https://wiki.opennms.org/wiki/Severity` (visited on 01/03/2018).

[65] E. Haleplidis; J. Hadi Salim & S. Denazis. “Towards a Network Abstraction Model for SDNs”. In: *Journal of Network and Systems Management* (2015).

[66] R. Olups. *Zabbix 1.8 Network Monitoring*. Tech. rep. Packet Publishing, 2010, pp. 1–16. URL: `https://sisis.rz.htw-berlin.de/inh2012/12423451.pdf`.

[67] A. Dalle Vacche & S. Kewan Lee. *Mastering Zabbix*. Tech. rep. Packet Publishing, 2013, pp. 1–16. URL: `http://www.omid-online.com/ebooks/MasteringZabbix.pdf`.

[68] W.River. “Device Management in Internet of Things”. In: *Why It Matters and How to Achieve It* (2017).

[69] Z. Liu; F. Liu & K. Lin. “Agent-Based Device Management in RFID Middleware”. In: *4th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM’08)* (2008).

[70] S. Bera; S. Misra; S. K. Roy & M. S. Obaidat. “Soft-WSN: Software-Defined WSN Management System for IoT Applications”. In: *IEEE Systems Journal* (2016).

[71] B. Greevenbosch; K. Li & P. Van der Stok. *Candidate Technologies for COMAN*. IETF, Nov. 2017. URL: `http://tools.ietfs.org/html/draft-greevenboschcoman-candidate-tech-03` (visited on 01/03/2018).

[72] H. Lamaazi; N. Benamar; A. Jara; L. Ladid & D. El Ouadghiri. “Internet of thing and networks management: Lnmp, snmp, coman protocols”. In: *Int. Work. Wirel. Networks Mob. Commun.(WINCOM 2013)* (2013).

[73] Z. Sheng; H. Wang; C. Yin; X. Hu; S. Yang & V. C. Leung. "Lightweight management of resource-constrained sensor devices in internet of things". In: *IEEE internet of things* (2015).

[74] M. Castro; A. J. Jara & A. F. Skarmeta. "Enabling end-to-end coap based communications for the web of things". In: *IEEE internet of things* (2016).

[75] Open Mobile Alliance. *OMA Device Management Standardized Objects*. Tech. rep. Open Mobile Alliance, 2016, pp. 1–16. URL: `http://www.openmobilealliance.org/release/DM/V1_3-20100525-C/OMA-TS-DM_StdObj-V1_3-20100525-C.pdf`.

[76] Open Mobile Alliance. *OMA DM Device Description Framework*. Tech. rep. Open Mobile Alliance, 2012, pp. 1–34. URL: `https://resources.solitonsystems.com/manuals/DMES/4_0/5212.htm`.

[77] Open Mobile Alliance. *OMA Device Management Tree and Description*. Tech. rep. Open Mobile Alliance, 2012, pp. 1–14. URL: `http://www.openmobilealliance.org/release/DM/V1_3-20121009-C/OMA-TS-DM_TND-V1_3-20121009-C.pdf`.

[78] Open Mobile Alliance. *OMA DM Device Description Framework*. Tech. rep. Open Mobile Alliance, 2016, pp. 1–34. URL: `http://www.openmobilealliance.org/release/DM/V1_3-20160524-A/OMA-TS-DM_Protocol-V1_3-20160524-A.pdf`.

[79] N. Chu; D. Raouf; B. Corlay; M. Ammari; N. Gligoric; S. Krco; N. Ognjanovic & A. Obradovic. "OMA DM v1.x compliant Lightweight Device Management for Constrained M2M devices". In: *European Transactions on Telecommunications* (2013).

[80] V. Yadav; M. Verma & Nisha. "A Survey Paper on Wireless Access Protocol". In: *International Journal of Computer Science and Information Technologies, May 30, 2014*. IEEE. 2015.

[81] H. Derhamy; J. Eliasson; J. Delsing & P. Priller. "A Survey Paper on Wireless Access Protocol". In: *IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA), December 30, 2015*. IEEE. 2015.

[82] G. Diviney. "An Introduction to Short-Range Wireless Data Communications". In: *Embedded Systems Conference* (2003).

[83] M. Choi; J. Won-Ki Hong & J. Kim J. Kang; H. Ju. "OMA DM-based remote software fault management for mobile devices". In: *International Journal of Network Management* (2009).

[84] Z. Shelby; S. Akhouri & J. Höller G. Klas; F. Rodermund. *Lightweight M2M: Enabling Device Management and Applications for the Internet of Things*. Tech. rep. Ericsson and T-Mobile, 2014, pp. 1–21. URL: `https://www.omaspecworks.org/wp-content/uploads/2018/10/Whitepaper-11.1.18.pdf`.

[85] M. C. Ocak. "Implementation of an Internet of Things Device Management Interface". In: *Master Thesist* (2014).

[86] C. A. L. Putera & F. J. Lin. “Incorporating OMA Lightweight M2M protocol in IoT/M2M standard architecture”. In: *2nd World Forum on Internet of Things (WF-IoT)* (2015).

[87] S. Rao; D. Chendanda; C. Deshpande & V. Lakkundi. “Implementing LWM2M in constrained IoT devices”. In: *IEEE ICWiSe* (2015).

[88] Red Band Software. *FOTA Usage in the United States*. Tech. rep. ITU, 2011, pp. 1–7. URL: `https://www.itu.int/en/ITU-T/extcoop/cits/Documents/Meeting-201512-Arlington/007%5C%20-%5C%20Secure%5C%20Over-the-Air%5C%20Vehicle%5C%20Software%5C%20Updates%5C%20-%5C%20Operational%5C%20and%5C%20Functional%5C%20Requirements.docx`.

[89] S. N. Chowdhury; K. M. Kuhikar & S. Dhawan. “IoT Architecture: A Survey”. In: *International Journal of Industrial Electronics and Electrical Engineering, May 23, 2015*. Vol. 3. ijieee. 2015.

[90] G. Dai. “Design and implementation on SOAP- based things management protocol for internet of things”. In: *10th World Congress Intelligent Control and Automation (WCICA)* (2012).

[91] Wisys Technologies. *EZLux – Smart Street Lighting*. IETF, Apr. 2017. URL: `https://wisystech.com/solutions/ezlux-smart-street-lighting-bangalore/` (visited on 01/13/2018).

[92] B. A. G. Hillen; I. Passchier; E. F. Matthijssen; F. T. H. den Hartog & F. Selgert. “Remote management of mobile devices with broadband forum’s TR-069”. In: *IEEE Telecommunications Network Strategy and Planning Symposium* (2008).

[93] B. A. G. Hillen; I. Passchier; E. F. Matthijssen; F. T. H. den Hartog & F. Selgert. *TR-69 CWMP v1.4 Especifications*. IETF, July 2017. URL: `https://www.broadband-forum.org/technical/download/TR-069.pdf` (visited on 01/13/2018).

[94] & J. Song S. Husain; A. Prasad A. Kunz; A. Papageorgiou. “Recent Trends in Standards Related to the Internet of Things and Machine-to-Machine Communications”. In: *IEEE Telecommunications Network Strategy and Planning Symposium* (2014).

[95] K. Lee; H. Chu; J. Chu; Y. Lin; C. Hsiao & T. Hou. “ACS management capacity enhancement mechanism in CWMP”. In: *Electronics Letters* (2014).

[96] B. A. G. Hillen; I. Passchier; E. F. Matthijssen; F. T. H. den Hartog & F. Selgert. *Roll Out New Services with TR-069: A Cable IPTV Use Case*. Incognito, Sept. 2017. URL: `http://www.incognito.com/wp-content/uploads/cable-tr-069-iptv-use-case` (visited on 01/13/2018).

[97] T.-H. Wang; Y. C. Chen; C. M. Hsu; K. S. Hsu & H. C. Young. “Auto scaling of containerized ACSs for CPE management”. In: *Network Operations and Management Symposium (APNOMS)* (2014).

[98] M. Santos; T. O. Castro; D. F. Macedo & B. Horizonte. “ManIoT : Uma Plataforma para Gerenciamento de Dispositivos da Internet das Coisas”. In: *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos* (2016).

[99] K. Stravoskoufos; S. Sotiriadis; & E. Petrakis. "Iot-a and fiware: bridging the barriers between the cloud and iot systems design and implementation". In: *6th international conference on cloud computing and services science (CLOSER 2016)* 2 (2016).

[100] FIWARE. *FIWARE iot stack*. FIWARE Foundation, Apr. 2017. URL: `https://www.fiware.org/` (visited on 02/13/2018).

[101] FIWARE. *FIWARE iot stack*. FIWARE Foundation, Apr. 2017. URL: `http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Main%5C_Page` (visited on 02/13/2018).

[102] FIWARE. *FIWARE iot stack*. FIWARE Foundation, Apr. 2017. URL: `http://map.fiware.org/actors/smes/434` (visited on 02/13/2018).

[103] W.-G. Chang & F. J. Lin. "Challenges of incorporating oma lwm2m gateway in m2m standard architecture". In: *International conference Standards for Communications and Networking (CSCN)* (2016).

[104] S. K. Datta & C. Bonnet. "A lightweight framework for efficient m2m device management in onem2m architecture". In: *International Conference Recent Advances in Internet of Things (RIoT)* (2015).

[105] P. Jacobs; F. Ennesser & J. Song J. Swetina; G. Lu. "Toward a standardized common m2m service layer platform: Introduction to onem2m". In: *IEEE Wireless Communications* (2014).

[106] ONEM2M Standards for M2M and the Internet of Things. *Application Developer Guide - TR-0025 V1.0.0*. ONEM2M Foundation, Feb. 2017. URL: `https://www.etsi.org/deliver/etsi_tr/118500_118599/118525/01.00.00_60/tr_118525v010000p.pdf` (visited on 03/10/2018).

[107] ONEM2M Standards for M2M and the Internet of Things. *Application Developer Guide - Use Case*. ONEM2M Foundation, Feb. 2017. URL: `http://onem2m.org/application-developer-guide/use-case` (visited on 03/10/2018).

[108] SmartThing Project. *Use Case for SmartThings Project*. SmartThings Foundation, Feb. 2017. URL: `https://blog.smartthings.com/tag/smartthings-use-casesf` (visited on 03/10/2018).

[109] W. Qin; Q. Li; L. Sun; H. Zhu & Y. Liu. "Restthing: A restful web service infrastructure for mash-up physical and web resources". In: *IFIP 9th International Conference Embedded and Ubiquitous Computing (EUC)* (Mar. 2011).

[110] S. Lavanya. "A Smart Network: IoT to Monitor Temperature and Heart beat of a Person Using RFID Technology". In: *ECE Department, Aarupadai Veedu Institute of Technology, CHENNAI (T.N.) INDIA* (2016).

[111] P. P. Ray. "A survey of IoT cloud platforms". In: *Future Computing and Informatics Journal, July 10, 2015*. Vol. 3. Elsevier. 2017.

[112] R. Jang; W. Soh & S. Jung. "Design and Implementation of Data-Report Service for IoT Data Analysis". In: *International Conference on Chemical, Material and Food Engineering (CMFE-2015)* (2015).

[113] A. Wendel. *Customer Spotlight: Watts Water*. Xively Foundation, Apr. 2018. URL: `http://blog.xively.com/customer-spotlight-watts-water/` (visited on 07/10/2018).

[114] S. Lorenz. *Blueprint: A Central Control Hub for Connected Products*.

[115] Carriots. *Carriots–IoT Application Platform*. Carriots Foundation, Aug. 2018. URL: `https://www.carriots.com` (visited on 10/12/2018).

[116] M. Zdravkovic; M. Trajanovic; J. Sarraipa; R. Jardim-Gonçalves & M. Lezoche. "Survey of Internet-of-Things platforms". In: *6th International Conference on Information Society, Techology, ICIST, February 10, 2016*. Vol. 3. IEEE. 2016.

[117] A. Gluhak; O. Vermesan; R. Bahr; F. Clari; T. MacchiaMaria; T. Delgado; A. Hoeer; F. Bösenberg; M. Senigalliesi & V. Barchetti. *FOTA Usage in the United States*. Tech. rep. IoT EU Foundation, 2016, pp. 1–7. URL: `http://www.internet-of-things-research.eu/pdf/D03_01_WP03_H2020_UNIFY-IoT_Final.pdf`.

[118] J. A. Jara; M. A. Zamora & A. F. Skarmeta. "Knowledge acquisition and management architecture for mobile and personal health environments based on the Internet of things". In: *11th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)* (2012).

[119] IFTTT. *IFTTT - Connect the apps you love*. IFTTT Foundation, July 2018. URL: `https://ifttt.com/` (visited on 09/22/2018).

[120] Neha; M. S. Meena & Rajbir. "Implementation of SNMP (Simple Network Management Protocol) on Sensor Network". In: *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)* (2016).

[121] H. Hui-ping; X. Shi-de & M. Xiang-yin. "Applying SNMP Technology to Manage the Sensors in Internet of Things". In: *The Open Cybernetics & Systemics Journal* (2015).

[122] B. Hedstrom; A. Watwe & S. Sakthidharan. "Protocol Efficiencies of NETCONF versus SNMP for Configuration Management Functions". In: *PhD thesis* (2011).

[123] M. A. Marotta; C. B. Both; J. Rochol; L. Z. Granville & L. M. R. Tarouco. "Evaluating Management Architectures for Internet of Things Devices". In: *IEEE - Wireless Days (WD), 2014 IFIP* (2014).

[124] J. C. Silva; F. Andery; D. Mazzer & L. D. P. Mendes. "Factorial Design Analysis Applied to the Performance of Transmission Power Optimization Techniques for Wireless Sensor Networks". In: *XXII Iberchip Workshop* (2016).

[125] Stanford University. *TinyOS Documentation Wiki*. Stanford University, May 2018. URL: `http://tinyos.stanford.edu/tinyos-wiki/index.php/TinyOS_Documentation_Wiki` (visited on 10/12/2018).

[126] Contiki OS. *Contiki: The Open Source OS for the Internet of Things*. Contiki Foundation, Aug. 2018. URL: `http://www.contiki-os.net/` (visited on 10/12/2018).

[127] Momote.io. *Momote 001 - Design Documentation*. Momote Foundation, Aug. 2018. URL: `http://momote.io/momote001-en.html` (visited on 10/12/2018).

[128] Texas Instruments. *CC2538 - A Powerful System-On-Chip for 2.4-GHz IEEE 802.15.4-2006 and ZigBee Applications*. Texas Instruments, Aug. 2018. URL: `http://www.ti.com/product/CC2538` (visited on 10/12/2018).

[129] J. Strassner. “Policy-based Network Management: Solutions for the Next Generation”. In: *IEEE Communications Magazine* 42 (2014).

[130] J. Van der Ham. “Challenges of an information model for federating virtualized infrastructures”. In: *5th International DMTF Academic Alliance Workshop on Systems and Virtualization Management: Standards and the Cloud* (2011).

[131] J. Jiyong; H. Saeyoung; K. Jinseok; P. Sungyong; B. Seungjo & C. W. Young. “7th IEEE International Conference on Computer and Information Technology”. In: (2011).

[132] Van Deursen and Van Dijk. “Using the Internet: Skill related problems in users online behavior”. In: *Interacting with Computers* (2009).

[133] Tools Moz SEO Software. *Resources for Smarter Marketing*. Moz SEO Software, Aug. 2018. URL: `https://moz.com/` (visited on 10/12/2018).

[134] Alexa. *Website Traffic, Statistics and Analytics*. Alexa, Oct. 2018. URL: `http://www.alexa.com/siteinfo` (visited on 11/22/2018).

[135] PMI – PROJECT MANAGEMENT INSTITUTE. 2003.

[136] W3Schools. *XML Tutorial*. Alexa, Aug. 2018. URL: `https://www.w3schools.com/xml/` (visited on 09/22/2018).

[137] ERCIM. “Network description tools and standards”. In: *Future Internet Technology* (2009).

[138] PM.U. Farooq; Muhammad Waseem; Anjum Khair & Sadia Mazhar. “A Critical Analysis on the Security Concerns of Internet of Things (IoT)”. In: *International Journal of Computer Applications* (2015).

[139] University of Amsterdam SNE research group. *Network Description Language*. University of Amsterdam SNE research group, Aug. 2018. URL: `http://www.science.uva.nl/research/sne/nd` (visited on 09/22/2018).

[140] R. K. Ghosh. “Wireless Networking and Mobile Data Management”. In: *Communication Networks* (2017), pp. 1–67.

[141] Apache Software Foundation. *Apache JMeter: load test functional and measure performance*. Apache Software Foundation, Nov. 2018. URL: `http://jmeter.apache.org/` (visited on 12/22/2018).

[142] B. Hedstrom; A. Watwe & S. Sakthidharans. “Protocol Efficiencies of NETCONF versus SNMP for Configuration Management Functions”. In: *Masters in Interdisciplinary Telecommunications* (2011).

[143] R. Pressman. *Engenharia de Software - Uma Abordagem Profissional*. Bookman, 2016.

[144] J. Preece; Y. Rogers & H. Sharp. *Design de interação – Além da interação homem-computador*. Bookman, 2005.

[145] D. M. Fernandez; S. Wagner; K. Lochmann; & A. Baumann. “Field study on requirements engineering: Investigation of artefacts, project parameters, and execution strategies”. In: *Information and Software Technology* 54 (2012).

[146] K. Curcio; T. Navarro; A. Malucelli; & S. Reinehr. “Requirements engineering: A systematic mapping study in agile software development.” In: *Journal of Systems and Software* 154 (2018).

[147] IoT RG - Inatel. *In.IoT - a middleware platform for Internet of Things (IoT).* Inatel, July 2018. URL: `http://jmeter.apache.org/` (visited on 12/22/2018).

[148] Jonathan de C. Silva & Joel J. P. C. Rodrigues. *M4DN.IoT – Management for IoT Devices and Network.* Tech. rep. Instituto Nacional de Telecomunicações, 2018.

[149] M. Christoffersen. “Demand side management of electric water heater with a photovoltaic system”. In: *Master thesis* (2018).

[150] Google. *AngularJS — Superheroic JavaScript MVW Framework.* Google Foundation, Nov. 2018. URL: `https://angularjs.org/` (visited on 12/22/2018).

[151] R. Lerdorf.

[152] Corrente. *Node.js: a JavaScript engine.* NodeJS Foundation, Nov. 2018. URL: `https://nodejs.org/` (visited on 12/22/2018).

[153] Creative Commons. *REST API Tutorial.* Creative Foundation, Nov. 2018. URL: `https://www.restapitutorial.com/` (visited on 12/22/2018).

[154] M. Otto & J. Thornton. *Bootstrap: The most popular HTML, CSS, and JS library in the world.* MIT, Nov. 2018. URL: `https://getbootstrap.com/` (visited on 12/22/2018).

[155] A. Jacomy & G. Plique. *Sigma: a JavaScript library dedicated to graph drawing.* MIT, Nov. 2018. URL: `http://sigmajs.org` (visited on 12/22/2018).

[156] A. K. Shulman; W. Shulman & J. Cottrell. *MongoDB Hosting: Database-as-a-Service by mLab.* MongoDB, Nov. 2018. URL: `https://mlab.com/` (visited on 12/22/2018).

[157] SmartThing Project. *Developer Documentation: Release 1.0.* SmartThings Foundation, Feb. 2017. URL: `https://media.readthedocs.org/pdf/smartthings/latest/smartthings.pdf` (visited on 03/10/2018).

[158] Digital CX & IoT I Europe. “IoT Platforms for Device Management: Positioning of Bosch Software Innovations”. In: *SITSI I Vendor Analysis I PAC INNOVATION RADAR* (2017).

[159] M. Stusek; P. Masek; K. Zeman; D. Kovac; P. Cika; J. Pokorny & F. Kröpfl. “A Novel Application of CWMP: An Operator-grade Management Platform for IoT”. In: *International Journal of Advances in Telecommunications Electrotechnics, Signals and Systems* (2016).

[160] C. Thomson; I. Romdhani; A. Y. Al-Dubai & I. Wadhaj. *Cooja Simulator Manual.* Contiki Foundation, July 2018. URL: `https://anrg.usc.edu/contiki/index.php/Cooja_Simulator` (visited on 12/22/2018).