HAL
open science

Towards interoperability, self-management, and scalability for scalability for machine-to-machine systems

Mahdi Ben Alaya

To cite this version:

Mahdi Ben Alaya. Towards interoperability, self-management, and scalability for scalability for machine-to-machine systems. Networking and Internet Architecture [cs.NI]. INSA de Toulouse, 2015. English. NNT : 2015ISAT0052 . tel-02059727v1

HAL Id: tel-02059727

https://theses.hal.science/tel-02059727v1

Submitted on 6 Mar 2019 (v1), last revised 17 Aug 2016 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Institut National des Sciences Appliquées de Toulouse (INSA de Toulouse)*

Présentée et soutenue le *06/07/2015* **par :**

Mahdi BEN ALAYA

Towards Interoperability, Self-management, and Scalability for Machine-to-Machine Systems

JURY

Omar ELLOUMI	Alcatel-Lucent	Invité
Michael MRISSA	Univ. Claude Bernard Lyon 1	Rapporteur
Didier DONSEZ	Univ. Grenoble 1	Rapporteur
Luigi Alfredo GRIECO	École polytechnique de Bari	Examinateur
Christophe CHASSOT	INSA de Toulouse	Examinateur
Thierry MONTEIL	INSA de Toulouse	Codirecteur de thèse
Khalil DRIRA	LAAS-CNRS	Directeur de thèse

École doctorale et spécialité :
MITT : Domaine STIC : Réseaux, Télécoms, Systèmes et Architecture
Unité de Recherche :
LAAS - CNRS (UPR 8001)
Directeur(s) de Thèse :
Khalil DRIRA et *Thierry MONTEIL*
Rapporteurs :
Didier DONSEZ et *Michael MRISSA*

Acknowledgement

I wish to express my thanks to the director of LAAS-CNRS Dr. Jean Arlat for accepting me into his laboratory and providing an ideal environment for scientific research.

I would like to express my gratitude and sincere appreciation for my thesis director, Dr. Khalil Drira for allowing me to conduct my research under his auspices. I am especially grateful for his confidence and the freedom he gave me to do this work. His extensive knowledge, vision, and creative thinking have been the source of inspiration for me throughout this research and allowed me to grow as a researcher.

As a thesis supervisor, Dr. Thierry Monteil has been a tremendous mentor for me. He supported me in all stages of this work and gave me continuous encouragement and inestimable advices, despite his busy agenda. Our in-depth and infinite discussions have invaluably shaped the course of my research and led me to the right way.

I am deeply grateful to Dr. Luigi Alfredo Grieco who offered me valuable suggestions for this study and spent much time with me to improve my publications. My sincere appreciation is extended to Dr. Omar Elloumi who kindly helped me with a great efficiency and contributed to the success of my thesis.

I would like to acknowledge the assistance of Dr. Christophe Chassot who helped me to overcome several difficulties encountered during my research and gave me priceless advices for my career. Special thanks to the members of the jury Dr. Didier Donsez and Dr. Michael Mrissa for agreeing to read the manuscript, for their brilliant comments and suggestions, and for letting my defense be an enjoyable moment.

I particularly want to thank all the members of the SARA team at LAAS-CNRS including Samir Medjiah, Slim Abdellatif, Veronique Baudin, Said Tazi, Gene Cooperman, Jean Marie Garcia, Michel Diaz and all my friends and colleagues including Tom Guerout, Amine Hannachi, Aymen Kammoun, Mohamed Siala, Elvis Vogli, Amine Haj Kacem, Emna Mezghenni, CodÃĭ Diop, Yassine Banouar, Marouane Elkias, Maroua Meddeb, Yassine Feki, Guillaume Garzone, FranÃğois AÃŕssaoui, Nicolas Seydoux, ChloÃĭ Bazile, Salma Mattoussi, Marc Bruyere, Jorge R. Gomez, Sakkaravarthi Ramanathan, German Sancho, Hatem Arous, and Ismael Buassida. The friendship, enlightening discussions, and the overall atmosphere center spirit have made my stay at LAAS-CNRS pleasant and enjoyable.

This work is dedicated to my parents Ferid and Sarra, my sister Souha, my brother Walid, and all my family for their love and all the sacrifices they have made for me. This diploma is just as much yours as it is mine.

Finally, I would like to express my heartfelt thanks to my beloved wife Ghada who spent sleepless nights, taking care of me. She has always inspired me to do my best and has supported me in every decision I have made.

Abstract

Machine-to-Machine (M2M) is one of the main features of Internet of Things (IoT). It is a phenomenon that has been proceeding quietly in the background, and it is coming into the surface, where explosion of usage scenarios in businesses will happen. Sensors, actuators, tags, vehicles, and intelligent things all have the ability to communicate. The number of M2M connections is continuously increasing, and it has been predicted to see billions of machines interconnected in a near future. M2M applications provide advantages in various domains from smart cities, factories of the future, connected cars, home automation, e-health to precision agriculture. This fast-growing ecosystem is leading M2M towards a promising future. However, M2M market expansion opportunities are not straightforward. A set of challenges should be overcome to enable M2M mass-scale deployment across various industries including interoperability, complexity, and scalability issues.

Currently, the M2M market is suffering from a high vertical fragmentation affecting the majority of business sectors. In fact, various vendor-specific M2M solutions have been designed independently for specific applications, which led to serious interoperability issues. To address this challenge, we designed, implemented, and experimented with the OM2M platform offering a flexible and extensible operational architecture for M2M interoperability compliant with the SmartM2M standard. To support constrained environments, we proposed an efficient naming convention relying on a non-hierarchical resource structure to reduce the payload size. To reduce the semantic gap between applications and machines, we proposed the IoT-O ontology for an effective semantic interoperability. IoT-O consists of five main parts, which are sensor, actuator, observation, actuation and service models and aims to quickly converge to a common IoT vocabulary.

An interoperable M2M service platform enables one to interconnect heterogeneous devices that are widely distributed and frequently evolving according to their environment changes. Keeping M2M systems alive is costly in terms of time and money. To address this challenge, we designed, implemented, and integrated the FRAMESELF framework to retrofit self-management capabilities in M2M systems based on the autonomic computing paradigm. Extending the MAPE-K reference architecture model, FRAMESELF enables one to dynamically adapt the OM2M system behavior according to high level policies how the environment changes. We defined a set of semantic rules for reasoning about the IoT-O ontology as a knowledge model. Our goal is to enable automatic discovery of machines and applications through dynamic reconfiguration of resource architectures.

Interoperability and self-management pave the way to mass-scale deployment of M2M devices. However, current M2M systems rely on current internet infrastructure, which was never designed to address such requirements, thus raising new requirements in term of scalability. To address this challenge, we designed, simulated and validated the OSCL overlay approach, a new M2M meshed network topology as an alternative to the current centralized approach. OSCL relies on the Named Data Networking (NDN) technique and supports multi-hop communication and distributed caching

to optimize networking and enhance data dissemination. We developed the OSCLsim simulator to validate the proposed approach. Finally, a theoretical model based on random graphs is formulated to describe the evolution and robustness of the proposed system.

Contents

1 Introduction **15**

2 OM2M: Interoperable M2M service platform **23**
2.1 Introduction . . . 23
2.2 The vision of Internet of Things (IoT) . . . 24
2.3 The Machine-To-Machine (M2M) paradigm . . . 24
2.3.1 M2M definition . . . 24
2.3.2 M2M application domains . . . 25
2.3.3 M2M challenges . . . 26
2.4 Architecture styles . . . 28
2.4.1 Remote Procedure Call (RPC) . . . 28
2.4.2 Representational State Transfer (REST) . . . 29
2.5 M2M related works . . . 29
2.5.1 ETSI SmartM2M . . . 30
2.5.2 TIA TR-50 . . . 30
2.5.3 OneM2M . . . 31
2.5.4 OMA Service Enablers . . . 31
2.5.5 IETF ROLL & CORE . . . 31
2.5.6 IoT-A . . . 32
2.5.7 OpenIoT . . . 32
2.5.8 BETaaS . . . 32
2.6 SmartM2M service architecture . . . 33
2.6.1 High level system architecture . . . 33
2.6.2 Service capability framework . . . 35
2.6.3 Resource structure . . . 36
2.6.4 Interface procedures . . . 37
2.7 OM2M standard-based service platform for M2M interoperability . . . 38
2.7.1 OM2M platform overview . . . 38
2.7.2 SCL Resource tree . . . 38
2.7.3 Main building blocks . . . 40
2.7.4 Components diagram . . . 41
2.7.5 Dynamic discovery of services . . . 43
2.7.6 Protocol-independent CORE module . . . 43
2.7.7 Binding to multiple communication protocols . . . 45

2.7.8 Interworking with legacy devices . . . 47
2.7.9 Visualization interfaces . . . 51
2.8 Enabling constrained devices in SmartM2M . . . 52
2.8.1 SmartM2M hierarchical resource structure . . . 52
2.8.2 Proposed non hierarchical resource structure . . . 53
2.8.3 Resource naming conventions evaluation . . . 55
2.9 IOT-O ontology for semantic data interoperability . . . 56
2.9.1 Towards semantic IoT . . . 56
2.9.2 IoT-O ontology model . . . 57
2.9.3 Use cases . . . 57
2.10 Conclusion . . . 59

3 FRAMESELF: Self-managing M2M systems 61
3.1 Introduction . . . 61
3.2 Autonomic computing paradigm . . . 62
3.3 Self-management approaches . . . 63
3.3.1 Distributed databases . . . 63
3.3.2 Agent architectures . . . 64
3.3.3 Component based frameworks . . . 64
3.3.4 Artificial intelligence and control theory . . . 64
3.3.5 Service Oriented Architecture . . . 64
3.3.6 Autonomic infrastructures . . . 64
3.4 Self-management techniques . . . 65
3.4.1 Self-configuration . . . 65
3.4.2 Self-optimization . . . 65
3.4.3 Self-healing . . . 65
3.4.4 Self-protecting . . . 65
3.5 FRAMESELF: Autonomic M2M framework . . . 65
3.5.1 Component-based architecture . . . 66
3.5.2 Management data elements . . . 69
3.5.3 Visualization and administration Interface . . . 72
3.6 Home automation self-configuration use case . . . 72
3.6.1 Mapping infrastructure . . . 73
3.6.2 Home sensors monitoring . . . 73
3.6.3 Home environment analyzing . . . 76
3.6.4 New configuration planning . . . 77
3.6.5 Home actuators controlling . . . 77
3.7 Semantic matching and resource reconfiguration . . . 77
3.7.1 Mapping infrastructure . . . 78
3.7.2 M2M resources discovery . . . 79
3.7.3 Application and device semantic matching . . . 80
3.7.4 New connections planning . . . 81
3.7.5 Resource dynamic reconfiguration . . . 83
3.8 Conclusion . . . 85

4 OSCL: Information centric M2M network **87**
4.1 Introduction . . . 87
4.2 Network overlay in M2M systems . . . 88
4.2.1 Wireless Mobile and Sensor Networks . . . 88
4.2.2 Distributed Hash table (DHT) . . . 88
4.2.3 Host Identity Protocol (HIP) . . . 89
4.2.4 eXtensible Messaging and Presence Protocol (XMPP) . . . 89
4.2.5 Enhanced Dynamic Service Overlay Network (e-DSON) . . . 89
4.2.6 Information Centric Networking (ICN) . . . 89
4.2.7 Named Data Networking (NDN) . . . 90
4.3 Overlay Service Capability Layer . . . 91
4.4 OSCL example use cases . . . 93
4.5 OSCLsim: NDN simulator for OSCL . . . 95
4.5.1 OSCLsim features and interface . . . 96
4.5.2 OSCL performance evaluation . . . 96
4.6 Dynamic random graph model for diameter constrained networks . . . 99
4.6.1 Target Scenario and Notation . . . 99
4.6.2 Dynamics of The Average Graph Degree . . . 101
4.7 DRG model simulations . . . 103
4.8 DRG model example applications . . . 104
4.8.1 Rank bound and comparison . . . 104
4.8.2 Robustness . . . 106
4.8.3 Transient duration . . . 107
4.8.4 Link failures . . . 109
4.9 Conclusion . . . 109

5 Conclusion and perspectives **111**

List of Figures

2.1 SmartM2M High level architecture . . . 34
2.2 SmartM2M functional architecture . . . 36
2.3 OM2M high level architecture . . . 39
2.4 OM2M resource tree example . . . 40
2.5 OM2M building blocks . . . 41
2.6 OM2M components diagram . . . 42
2.7 OM2M plugins discovery . . . 43
2.8 Protocol-independent exchange structure . . . 45
2.9 Protocol-independent CORE module . . . 46
2.10 HTTP binding example . . . 47
2.11 Binding to multiple communication protocols . . . 48
2.12 Interworking with legacy devices . . . 49
2.13 oBIX XML description example . . . 50
2.14 oBIX XML data example . . . 50
2.15 Resource tree browser web interface . . . 51
2.16 Device monitoring web interface . . . 52
2.17 Comparison between Hierarchical and non Hierarchical URI . . . 54
2.18 Create Evaluation . . . 55
2.19 Retrieve evaluation . . . 56
2.20 Ontology model for M2M . . . 58
2.21 Temperature sensor model example using the M2M ontology . . . 59
2.22 Lamp actuator model example using the M2M ontology . . . 60

3.1 MAPE-K loop reference architecture . . . 63
3.2 FRAMESELF components diagram . . . 66
3.3 FRAMESELF data format . . . 70
3.4 FRAMESELF demonstration interface . . . 73
3.5 FRAMESELF administration interface . . . 74
3.6 Ontology model for M2M . . . 78
3.7 M2M ontology instance . . . 80

4.1 OSCL operations: simplified flow chart. . . . 92
4.2 A Simple ETSI M2M scenario. . . . 93
4.3 Message Sequence Chart with OSCL. . . . 94
4.4 An extended SmartM2M scenario with P2P capabilities. . . . 95

4.5 Message Sequence Chart with OSCL and ICN messages. 95
4.6 SmartM2M NDN simulator graphical interface 96
4.7 Evolution of the OSCL network 97
4.8 Transactions distribution without OSCL 97
4.9 Transactions distribution using OSCL 98
4.10 Transactions distribution using OSCL with caching 98
4.11 Evolution of a random graph ($N = 5$, $D = 2$) 99
4.12 Average absolute relative errors of the model (4.14). 103
4.13 Steady State absolute relative errors of the model (4.14). 104
4.14 Evolution of the number of edges over the time t, ($N = 1000$, $D = 5$). 105
4.15 Evolution of P_t, ($N = 1000$, $D = 5$). 105

List of Tables

2.1 Comparative table of interoperable M2M solutions 33

4.1 Notation. 100

Chapter 1

Introduction

The internet is one of the most important creations in all of human history. It is a massive networking infrastructure used to send information quickly between computers around the world. It is a network of networks composed of domestic, academic, business, and government networks linked by a broad array of wired and wireless technologies. It carries an extensive range of services, such as email, file transport, peer-to-peer sharing, video streaming, telephony, the World Wide Web (web), etc. The web is the most popular information sharing model built on top of the internet. It is a system of interlinked hypertext documents accessible via the internet that makes the flowing information usable. So in other words, the Internet is the physical layer made up of switches, routers, and other equipment, and the web is the application layer we use to access the Internet.

The internet and mainly the web have gone through several distinct evolutionary phases [1]. At the beginning, it was the research phase where the internet was just an experiment in reliable networking and an effort to link a large number of universities and research centers. Then, appeared the static web where almost every company registered a domain name and created a cheaply designed web site with no interaction used primarily to advertise its products or services. After that, we moved to the transactional web, where products and services could be bought and sold, and services could be delivered. Next, it was the social web, where companies like Facebook and Twitter have become immensely popular and profitable by allowing people to communicate, connect, and share information about themselves with friends, family, and colleagues. And now, we are moving to the Internet of Things (IoT), where small devices such as sensors, actuators, and radio-frequency identification tags, have been connected to the Internet, all sharing information with each other without human intervention.

IoT is the most important evolution of the Internet that have the potential to dramatically improve the way people live, learn, work, and entertain themselves. Semantically, the IoT means "a world-wide network of interconnected objects" uniquely addressable, based on standard communication protocols" [2]. However, a complete agreement on the definition is missing. Peter Hartwell, a senior researcher at HP Labs said: "With a trillion sensors embedded in the environment - all connected by computing systems, software, and services - it will be possible to hear the heartbeat of the Earth, impacting human interaction with the globe as profoundly as the Internet has revolutionized communication."

While IoT is relatively a new phenomenon that is changing the world in which we live, an old concept called Machine-to-Machine communications (M2M) has been proceeding quietly in the

background, and is coming into the surface. M2M is the communication paradigm that enables machines to communicate together without human intervention. It enables a device to exchange information with a business application via a communication network, so that the device or application can act accordingly based on exchanged information. In an attempt to explain the relationship between IoT and M2M concepts, Matt Hatton of Machina Research describes M2M as the "plumbing" of the IoT.

M2M provides benefits to many individuals, companies, communities and organizations in the public and private sectors, ranging from energy, building, health-care, environment, transport, retail, industrial to security and public safety. This fast-growing ecosystem is leading M2M towards a promising future, however M2M market expansion opportunities are not straight forward. In fact, several big challenges must be overcome in different aspects such as market vertical fragmentation, extensibility, increasing complexity, semantic gap, power management, network misalignment, scalability and security [3].

Research problems

M2M is suffering from the highly fragmented vertical domain specific approach, which has increased the R&D cost in each specific domain. The lack of a horizontal M2M service platform that provides both communication and semantic interoperability and that can work in constrained environments is a real handicap to realize the vision of IoT. At the same time, M2M communications are set for a massive growth that will reach a level of complexity where the human effort required to get the systems up and running is getting out of hand. In addition, M2M applications have given rise to new requirements from the internet infrastructure, such as support for scalable content distribution, however the Internet was never designed to address such requirements which is a barrier for M2M mass scale deployment.

M2M domain vertical fragmentation

Various vertical M2M solutions have been designed independently and separately for different applications, which inevitably impacts M2M interoperability and consequently impedes large-scale M2M deployment. The existence of multiple manufacturers, the lack of a common architecture for M2M, and no clarity about what can be achieved have all combined to leave the field of M2M communications closer to dream than reality.

- **Lack of communication interoperability:** To achieve M2M communication interoperability, it is required to standardize a service layer with open interfaces which is already done in many standards such as ETSI SmartM2M. However, the system architecture design that will bring life to such horizontal solution presents, in itself, many challenges. In fact, the system architecture must be flexible enough to be deployed in different kind of machines. Given that the platform cannot support from scratch all existing technologies and protocols, the architecture must be modular, highly extensible and support service dynamic discovery.

- **Lack of semantic interoperability:** Achieving communication interoperability is not sufficient to solve the semantic gap between machines required for machine computable logic, inferencing, knowledge discovery, and data federation. Actually, there is no common dictionary describing formally relevant M2M concepts, attributes and relations without ambiguity.

In fact, each M2M application or device must personalize the payload by using its own vocabulary, so all interacting applications must agree beforehand on a specific terminology before establishing any communications. This implies necessarily a strong coupling between applications which is in contradiction with M2M concept of horizontality.

- **Difficulty to adapt to constrained environments:** The vast majority of devices that will constitute the IoT are expected to be severely constrained in terms of memory, CPU, and power capacities. Unfortunately, the integration of these constrained devices into a horizontal service platform is not straightforward. M2M systems must address a large number of resources with complex relationships which result in long resource URIs. In addition, it must enable meta-data exchange between applications and devices which results in big resource representation. Even when using a dedicated constrained protocol, an inefficient naming convention and a not optimized resource structure will compromise the mass deployment of M2M devices.

M2M increasing complexity

A similar problem of increasing complexity was experienced in the 1920s in telephony where human operators were required to work the manual switchboards [4]. Because the usage of the telephone increased rapidly, there were serious concerns with insufficient number of trained operators to work the switchboards.

- **Absence of self-management capabilities:** An interoperable platform makes it easy to interconnect heterogeneous devices and applications in one M2M system and makes it possible to connect various M2M systems together. As a result, such growing number of interconnected machines is going to increase the administration overhead for the management of these large systems. Management operations are costly in terms of time and money and require a permanent presence of high skilled administrator. Despite the criticality of this challenge, current horizontal M2M solutions do not define any mechanism to provide M2M systems with self-management capabilities to reduce unnecessary human intervention.

- **No support for semantic-based reasoning:** M2M systems enable to collect, store and share large and complex data coming from multiple sources. The quality and consistence of the data being captured can vary greatly hampering the process of data management process. This data need to be linked and correlated in order to be able to grasp the information that is supposed to be conveyed by this data. A semantic-based reasoning mechanism is missing in current horizontal M2M solutions to enable device dynamic discovery or to determine relevant information or critical action that has not been explicitly told about.

M2M deployment unscalability

In order to support new M2M application requirements, the internet undergone a vicious cycle of functionality patches, such as Mobile IP. Most of those patches increased the complexity of the overall architecture and proved to be only temporary solutions. This has raised the question of whether we can continue “patching over patches” the internet or whether a new clean-slate architectural approach for the Internet is actually needed.

- **Inefficient networking and data distribution:** In general, M2M solutions rely on a centralized structure organization in which the M2M server is responsible for handling and relaying all distributed signaling in the system. It is well known that centralized systems can surely yield strong optimization but, at the same time, critical entities of the system that act as concentrators, such as the M2M network, may suddenly become single points of failure, thus lowering the fault tolerance of the entire system. Even if the M2M server can be implemented in a distributed way on the cloud, its scale and costs would be much higher if the capabilities of distributed gateways and devices are not fully exploited.

- **Difficulty in dimensioning and monitoring the system behavior:** M2M enables to build complex networked systems, made of interacting dynamic units. However the properties of such interacting units cannot be deterministically known in advance. The dimensioning process involves determining the M2M network topology, routing plan, traffic matrix, transient duration, and robustness and reliability requirements, and using this information to determine the maximum call handling capacity of the equipment, and the maximum number of connections required between the machines. Such process requires a complex model that simulates the behavior of the network equipment and routing protocols.

Contributions

The main goal of this thesis is to go towards a horizontal M2M service platform that provides cross-domain interoperability, self-management capabilities, and deployment scalability. Interoperability means the ability of M2M applications and devices to exchange data in a seamless way using communication protocol and semantic data even in constrained environments. Self-management denotes the capacity of the M2M system to maintain and adjust its functioning to face workloads, demands, and environment changes according to high level policies and without human intervention. Deployment scalability means the ability for M2M systems to be enlarged and sized to accommodate to a growing number of devices or requests.

M2M cross-domain interoperability

To reduce the standardization gap, which exists between M2M domains, many standards like ETSI SmartM2M [3], TIA TR-50 [5], OneM2M [6], OMA service enabler [7], or IETF CORE and ROLL [8], and research projects like IoT-A [9], OpenIoT [10], or BeetTAAS [11] addressed the need for a common M2M service platform. After deeply studying cited solutions [12], it appears that there is no solution that overcomes all targeted challenges. However, ETSI SmartM2M remains the most mature and complete one. The SmartM2M standard is our starting point to build an end-to-end M2M service platform with the intermediate service layer that are key components of the horizontal M2M solution.

- **OM2M: extensible SmartM2M-based service platform for M2M interoperability** We designed and developed the OM2M platform [13], a SmartM2M-based M2M service platform to enable interoperability between vertical applications in different domains. It offers a standardized RESTful API that enables applications and machines to perform complex scenarios based on primitive procedures independently of the underlying network. It is designed to be flexible in order to be easily deployed in different type of machines from servers,

gateways, to devices. OM2M proposes a modular architecture, extensible via plugins. The OM2M core plugin is independent of any communication protocol and any device technology. Additional plugins are added to extend OM2M features to bind to different communication protocols, interwork with a variety of legacy technologies, and interface with existing device management protocols. OM2M is provided as the first fully open source implementation [1] of the SmartM2M standard and is disseminated through the Eclipse Foundation [2].

- **Non hierarchical naming convention for constrained environments** We proposed an efficient naming convention [14] based on a non hierarchical resource structure for SmartM2M and experimented in OM2M to meet the requirements imposed by constrained devices. The main idea behind the proposed solution consists in defining a new URI template for all resources in such a way that they become addressable in a short and homogeneous way. The new template limits the number of nested resources and therefore optimizes the overall URI length. It is sufficiently generic to cover all the primitive procedures defined in SmartM2M and make it work even in constrained environments.

- **IoT-O: IoT Ontology for semantic data interoperability** The IoT-O ontology [15] is proposed to enable semantic interoperability in M2M system in general and SmartM2M standard in particular. IoT-O represents devices meta-data, operations, and exchanged data independently from the underlying formalism originally used for describing them. Since ontologies are designed to be reusable and extensible, the main idea of our approach consists in creating a complete ontology for IoT by reusing existing ontologies. New concepts are designed only when needed. IoT-O consists of five main parts which are sensor, observation, actuator, actuation and service models and aims to reduce the ambiguity of IoT terminology and allows one to converge quickly to a common vocabulary.

M2M self-management capabilities

To reduce the M2M increasing complexity, the autonomic computing paradigm was introduced in 2001 to improve computing systems in order to decrease human involvement [16]. The term "autonomic" comes from biology. In the human body, the autonomic nervous system takes care of unconscious reflexes. It governs our heart rate, body temperature, pupil size, and respiration rate thus freeing our conscious brain from the burden of dealing with these and many other low-level, yet vital, functions [17]. An autonomic manager is organized into four main modules: monitor, analyzer, planner and executor. These modules share a same knowledge, exploit high level policies, and constitute together the MAPE-K control loop [18] for self-managing entities using sensors and actuators.

- **FRAMESELF: Autonomic manager architecture for self-managing M2M systems** We designed, developped and integrated the FRAMESELF framework [19–23] to retrofit self-management capabilities in M2M system to overcome their increasing complexity. The main goal here is to consolidate existing efforts and provide a solution that works in a multi-vendor environment. FRAMESELF proposes a concise and modular autonomic manager architecture that extends the current MAPE-K loop reference model and exploits existing technologies and standards. It is integrated into OM2M as a new service to adapt the system behavior according to the environment changes.

[1] Eclipse OM2M project website: https://www.eclipse.org/om2m
[2] Eclipse OM2M press release: https://www.eclipse.org/org/press-release/20150408_om2m.php

- **Semantic reasoning for dynamic discovery and architecture reconfiguration** We defined a set of semantic rules [15, 19, 20, 24, 25] to enable FRAMESELF to reason on IoT-O as a knowledge model for dynamic reconfiguration of OM2M resource architecture. The main goal is to enable any application to dynamically discover interesting devices and to exchange data using the right communication mode according to its description, role, and relationships. This approach enables to avoid all repetitive and manual operations required by an applications to find relevant devices, which are pretty complex especially in a highly distributed environments.

M2M mass-deployment scalability

The use of network overlays in M2M systems has been widely studied in literature to improve networking and data dissemination, thus enhancing scalability. Different aspects are addressed including Wireless Mobile and Sensor Networks [26–28], distributed Hash table (DHT) [29, 30], Host Identity Protocol (HIP) [31], and Information-Centric Networking (ICN) [32]. ICN has emerged as a promising candidate for the architecture of the future Internet. By naming information at the network layer, ICN favors the deployment of in-network caching and storage and multicast mechanisms, thus enhancing overall system scalability. Many ICN architectures are available nowadays where stands the Named Data Networking (NDN) [33] that relies on hierarchical content names, receiver initiated sessions, content level security schema, and in-network caching mechanisms.

- **OSCL: Meshed M2M network for efficient networking and data dissemination** We proposed a new meshed network topology [34] for SmartM2M as an alternative to the current centralized approach to provide a more scalable solution. An Overlay of Service Capability Layers (OSCL) that relies on the emerging ICN paradigm is defined. OSCL supports multihop communication and distributed caching to enhance the efficiency of networking and data dissemination in complex scenarios. The NDN architecture is considered since it relies on communication primitives that could be easily integrated within the SmartM2M resource architecture. An NDN simulator for OSCL called OSCLsim is designed and developed to validate the proposed approach.

- **Dynamic random graph model for diameter constrained topologies** Graph-based models can be considered as fundamental tools to assess, predict, and control the performance of complex and dynamic networked systems. We formulated a theoretical model [35] based on random graph, which considers a discrete time process of arrivals to describe the sequence of vertex couples among, which a path composed of no more than D edges has to be established. Accordingly, a general topology formation mechanism is formulated, expressing the rules that drive the addition of new edges, obeying to the constrained on the maximum diameter D. Then, exploiting the properties of the binary adjacency matrix in graph theory, an original and tractable discrete time model is proposed that describes the evolution of the average degree of the corresponding network. The network robustness, transient time, and link failure are described as well.

Manuscript outline

The rest of this thesis is organized as follows:

Chapter 2. OM2M: Extensible SmartM2M service platform for an effective communication and data interoperability

This chapter provides a state of the art of M2M communications. Starting from the vision of IoT, it introduces the M2M paradigm and highlights its application domains, challenges, and architecture styles. Related works addressing the interoperability issues are discussed including current standardization activities, research studies. The OM2M platform operational architecture and main features are explained. The OM2M core module, service discovery mechanism and possible extensions are described. The inefficiency of SmartM2M naming convention to deal with constrained environments is highlighted and a new approach based on a non hierarchical resource template is presented and compared with the current approach. The IoT-O ontology proposal for semantic data interopeability is illustrated, and the main concepts and relationship of IoT-0 are presented and illustrated via real use cases.

Chapter 3. FRAMESELF: Autonomic Computing service for SmartM2M dynamic reconfiguration and semantic reasoning

This chapter provides a state of the art of the Autonomic Computing paradigm. It discusses the existing approaches, techniques and related works conducted to provide systems with self-management capabilities. The FRAMESELF autonomic manager architecture and data model are described. Two concretes use cases explaining how FRAMESELF can be adapted to different scenarios are provided. The first use case illustrates the use of simple knowledge and syntactic rules for the self-optimizing the home energy consumption. The second use case illustrates the use of ontology model and semantic rules for applications dynamic discovery and self-configuration of the M2M resource architecture.

Chapter 4. OSCL: Information Centric Networking in meshed SmartM2M architecture for efficient data dissemination

This chapter presents a state of the art about network overlays and summarizes the main features of the ICN and NDN architectures. The OSCL solution, which offers a meshed M2M network and makes use of the NDN architecture for an efficient networking and data dissemination is described and significant use cases are illustrated. An NDN simulator for OSCL called OSCLsim is designed and developed to validate the proposed approach. A dynamic random graph (DRG) model for diameter constrained topologies enabling to assess, predict, and control the performance of the OSCL network is presented, simulated, and validated and useful examples of its applications are described.

Chapter 5. Conclusion and perspectives

This last chapter provides an overview of all the work carried out and developed during this thesis, presents perspectives of extension and improvement, and gives an outlook into new research areas to follow up this work.

Chapter 2

OM2M: Interoperable M2M service platform

Contents

2.1 Introduction 23
2.2 The vision of Internet of Things (IoT) 24
2.3 The Machine-To-Machine (M2M) paradigm 24
2.4 Architecture styles 28
2.5 M2M related works 29
2.6 SmartM2M service architecture 33
2.7 OM2M standard-based service platform for M2M interoperability . . 38
2.8 Enabling constrained devices in SmartM2M 52
2.9 IOT-O ontology for semantic data interoperability 56
2.10 Conclusion 59

2.1 Introduction

In this chapter, we present the vision of IoT according to different definitions, and we introduce the M2M paradigm and compare it to IoT to point the difference between these two confusing concepts. Then, we present a set of interesting application domains where M2M brings benefits and highlight the most relevant requirements and challenges for. Next, we describe and compare the most popular architecture styles including RPC and REST to select the most appropriate style for our solution. After that, we present related works addressing the interoperability issues and discuss their horizontality, architecture style, communication protocols, semantic capabilities, and constrained environments support.

As a solution, we present first the OM2M platform that offers a flexible standard-based architecture for M2M cross-domain interoperability that can be deployed in different type of machines. OM2M extensibility, service dynamic discovery and main plugins including OM2M core, communication bindings, and interworking proxies are illustrated using UML components and sequence

diagrams. Second, we present a new resource naming convention based on a non hierarchical resource structure template to reduce the payload size for better support constrained environments. Finally, we presents the IoT-O ontology for semantic data interoperability. The main concepts and relationships of the proposed ontology are presented and illustrated via real use cases.

2.2 The vision of Internet of Things (IoT)

IoT has been a subject of intense discussions over the past five years. Some see it as the next technology revolution after the computer and Internet. Some consider it simply hype. Others are cautious with a wait-and-see attitude. IoT may refer to a transition state, an everyday object, a virtual identity, an extension of the internet, an enabling framework, or simply an interconnected world as the definition changes with relation to the speaker point of view.

According to Cisco, IoT was born when more "things" were connected to the Internet than people. The explosive growth of smartphones and tablet PCs brought the number of devices connected to the Internet to 12.5 billion in 2010. Refining these numbers further, Cisco IBSG estimates IoT was born sometime between 2008 and 2009 [1].

According to IoT Special Interest Group (SIG), IoT refers to a transition state where things will have more and more information associated with them and may have the ability to sense, communicate, network and produce new information, becoming an integral part of the Internet [36].

In the viewpoint of US National Intelligence Council (NIC), IoT refers to everyday objects that are readable, recognizable, locatable, addressable, and/or controllable via the Internet. It can focus on the virtual identity of the smart objects and their capabilities to interact intelligently with other objects, humans and environments [37].

According to IETF, IoT is an extension of internet technologies to constrained devices, moving away from proprietary architectures and protocols [38].

In [39], IoT can be understood as an enabling framework for the interaction between a bundle of heterogeneous objects and also as a convergence of technologies.

According to SAP, IoT is a world where physical objects are seamlessly integrated into the information network and where the physical objects can become active participants in business processes [40].

While IoT is relatively a new phenomenon that is changing the world in which we live, an old concept called M2M has been proceeding quietly in the background and came into the surface. Buzzwords can sometimes lead to confusion and cause questions to arise. So, are IoT and M2M the same thing? What is the difference between them?

2.3 The Machine-To-Machine (M2M) paradigm

In this section, we present several definitions of M2M and point the one to be used in this thesis. We describe also most relevant M2M application domains and highlight main challenges.

2.3.1 M2M definition

Many attempts have been made to propose a single and clear definition for M2M. Defining the complete M2M paradigm is not an easy task because the scope of M2M is elastic and the bound-

aries are not always clearly defined. Some use IoT and M2M interchangeably, whereas others are convinced that they are not to be confused.

M2M could be understood as a subset of IoT connecting devices to other devices using an Information and Communications Technology (ICT) system, whereas IoT goes a step beyond that and deals with things connected to other devices, people or systems. An example of this is in the supermarket where radio-frequency identification (RFID) tagged objects are offered to the customer. These objects are passive and have no direct means with which to communicate upstream with the M2M application but they can be read by an M2M scanner, which will be able to consolidate the bill, as well as making additional purchase recommendations to the customer. From this perspective, the M2M scanner is the "end point" of the M2M relationship [3].

M2M could be seen also as an industrial environment comprising of machines communicating on industrial protocols, and producing closed solutions, recently emerging into smart devices and appliances like refrigerators or energy meters. On the other hand, IoT is a broader concept comprising of applications pertaining to common usage, daily lives, possibly connecting things like garbage bins or books to the Internet and producing open solutions for mass.

M2M is also referred to as the "plumbing" of IoT, what provide connectivity that enables IoT services and capabilities. M2M involves connecting devices and transferring data, something that the IoT depends on and which would not be possible without it. In this thesis, we adopt a similar definition, M2M is the heart of IoT. However, we accept to use these two terms interchangeably.

2.3.2 M2M application domains

M2M provides benefits to many individuals, companies, communities and organizations in the public and private sectors, ranging from energy, building, health-care, environment, transport, retail, industrial to security and public safety.

- Energy: The Energy sector includes oil and gas energy extraction and transportation. It includes also renewable energy sources, such as wind, solar, wave, co-generation and electrochemical. M2M enhances power generation, transmission, and distribution for residential, commercial and industrial recipients with innovative technologies, varied regulations, changing price models, and decentralized power generation.

- Buildings: The Buildings sector includes HVAC, security and access, lighting, fire and safety systems that reside in buildings and facilities across commercial, institutional, industrial and residential segments. Smart buildings offer an increased level of automation. M2M allows innovative electrical appliances to communicate with intelligent power networks. Intelligent power meters monitor energy consumption and guide the usage of electricity, warm water, and heating.

- Industrial: The Industrial sector covers mechanical and plant engineering, production logistics, material handling, system assembly, industrial asset monitoring and tracking, version control, and location analysis for a wide range of factory processes. With M2M, companies gain a better overview of the condition of their equipment and can react quickly when something goes wrong.

- Healthcare: The Healthcare sector includes applications such as telemedicine, e-prescribing, electronic health records, collecting data using mobile devices, health knowledge management, and healthcare information system. Such applications will empower patients and physicians

alike to conduct better research and treatment options. This sector also includes Lab equipment, such as centrifuges, incubators, freezers and blood test equipment.

- Retail: The Retail sector covers networking systems and devices that allow retailers to have increased visibility of the supply chain, gather consumer and product information, quick wireless transactions, real-time reporting on inventory and sales data, and to reduce energy consumption. M2M allows merchants to tap into more sales opportunities by placing convenient, self-service machines in high-traffic areas. It enables customers to get immediate access to the items they want any time.

- Transport: The Transportation sector includes road, rail, air and sea transport as well as ticketing and passenger information systems, traffic management, parking space management and payment, traffic volume monitoring, connected road signs, traffic lights and enforcement cameras. This sector also includes transport systems and passenger information services, for example road pricing schemes and congestion charges and road tolls.

- Security: The Security sector includes business and home alarm systems, vehicle, asset, and animals tracking, people monitoring and safety solutions, food tracing, emergency and breakdown services and environmental monitoring. This sector also includes all military-related personnel and equipment, including special vehicles, weapons, surveillance and personnel systems.

2.3.3 M2M challenges

To accelerate the maturity of M2M solutions, several big challenges must be overcome in different domains despite progress in M2M technologies. The most important ones are:

- Fragmentation into vertical applications: Most industries are solving their M2M needs on their own. They are addressing specific vertical application requirements in isolation from each others despite similar architectures. The M2M "wheel" is reinvented from industry to industry. This has created "silo" solutions based on very heterogeneous forms of design, production, data model and implementation cycle [3]. Such unique solutions often result in vendor-specific hardware. Interoperability is in general very limited or non-existent. Development is limited to the system owners who understand the particular API, thus leading to high development costs and high costs for support. To overcome this challenge it is essential to define more comprehensive standards, in particular regarding communication interfaces and data models. In addition, it is important to have a common service platform that can be reused for multiple applications, avoiding the necessity to completely redesign solutions per application due to the lack of common capabilities.

- Extensibility: M2M is constantly changing and evolving. Unknown devices, applications, and use cases are added every day. M2M systems take their future growth into consideration by enabling developers to expand the system capabilities and facilitating systematic reuse. Extensions can be done through the addition of new functionalities or modification of existing ones while internal structure and data flow are minimally or not affected. To overcome this challenge, the M2M architecture design must be modular to enable building additional extension modules independently from the core project. Loose coupling of software and hardware components helps isolate problems when things go wrong and simplify testing, maintenance, and enhancement procedures.

- Complexity: The need to interconnect heterogeneous M2M devices and applications that have never been connected before and extend that to the internet introduced new levels of complexity. Soon M2M systems will become too massive and complex for even the most skilled developers and administrator to install, configure, maintain, and extend. To solve this challenge, the autonomic computing paradigm must be used to provide M2M system capable of self-management. Autonomic M2M systems will maintain and adjust their functioning to face workloads, demands, and environment changes, hardware or software failures, component upgrades, etc. Additionally, plug-and-play and zero configuration mechanisms must be considered to enable the average consumer to install and use M2M devices.

- Semantic gap: Humans are capable of using the web to carry out tasks such as reading an article, writing a comment, booking a hotel room, or interacting with other human users. Machines cannot accomplish all of these tasks without human direction, due to the large disparity between content descriptions that can be computed automatically. The current web is designed to be used by people, not machines. Addressing the semantic challenge requires a multi-disciplinary approach including computer science, cognitive science, signaling science, web science, and many others. Physical devices must be modeled using a higher abstraction layer and relevant information sources must be semantically structured to enhance interoperability. Reasoning with large amounts of data with ontological knowledge enables one to deal with all of vastness, vagueness, uncertainty, inconsistency, and deceit issues.

- Power management: A large number of M2M devices will be battery powered to be more portable and autonomous. The challenge is making it easy to add power management to these M2M devices. Rechargeable batteries are practical, but must tolerate more charge/discharge cycles. Many types of batteries employ toxic materials, which can be dangerous if not disposed of properly. Solar and human powered M2M devices can be a promising alternative by using clean renewable energies and human muscle power. Large M2M deployments require novel and stringent battery regulation, collection, recycling and safety programs to reduce the amount of released hazardous substances and protect the environment and human health.

- Network misalignment: The number of M2M devices generating very small amounts of data is growing drastically. Several references refer to this new non-voice communications growth as a "Data Tsunami" [41]. However, current communication networks are not designed for M2M applications especially that the behavior of M2M devices can be drastically different from human users. The introduction of an M2M application to an existing system will initiate rapidly many new subscribers causing very significant overload. Although some M2M devices may transmit intermittently, and may compete with one another and with other users impacting the network resources. To overcome this challenge the M2M traffic should be optimized, which will essentially take place within standardization.

- Security: M2M devices are typically required to be small, low cost, inexpensive, and sometimes mobile. They will be deployed in very large quantities and should be able to operate unattended by humans for extended periods of time. Additionally, standard operating systems and protocols are increasingly used in M2M applications. This means similar, well known, weaknesses are present in a large number of M2M devices and networks. The nature of M2M deployments introduces a number of unique security vulnerabilities such as physical attacks, theft and tampering of the M2M device, core network attacks, compromise of credentials, user data and identity privacy attacks, etc [42]. The traditional and global enforcement of

security will not be practical. To overcome this challenge, a trusted M2M environment must be established to avoid fraud, cyber-terrorism and criminality. Also, new challenging mechanisms should be invented to ensure that M2M devices can operate, locally, in a secure state without network connectivity.

2.4 Architecture styles

The software architecture of a program or computing system is the structure or structures of the system, which comprises software components, the externally visible properties of those components, and the relationships among them [43]. An architectural style defines a family of systems in terms of a pattern of structural organization, and a vocabulary of components and connectors, with constraints on how they can be combined [44]. In this section, two architecture styles are identified for M2M systems: Remote Procedure Call (RPC) [45] and Representational State Transfer (REST) [46]. REST and RPC were widely and deeply discussed in literature [47–49]. In this section, we introduce,and analyze these two architecture styles and resume the pros and cons of each one from the perspectives of interoperability, performance, complexity, and scalability.

2.4.1 Remote Procedure Call (RPC)

The RPC architecture style was invented during the distributed objects movement, which decades ago replaced in-memory object message exchange in object-oriented applications. RPC looks to the server as a set of procedures and the client calls these procedures to perform a specific task. RPC relies on procedures, which are verbal. Therefore the modeling based on RPC is verb-centralized, which is actually a kind of transaction script [50].

In RPC, a service is composed of small operations and has different interfaces. The interface contract is critical to service definition because it contains specific operating parameters and semantics. The semantics of each interface contract must be learned before using the service. This approach can more or less work in a closed application environment. However, when dealing with unforeseeable number of M2M devices within an open distributed system, this approach may cause problems of tight coupling and interface complexity, which impede scalability.

Service contract architecture defines also the data formats involved in interaction. For example, in SOA [51] we generally uses XML Schema to define data formats in WSDL. Any change in the contract requires all clients to be recompiled to adapt to the new definition. This approach makes the client and the server tightly coupled and does not help separating interface from implementation. Tightly coupled applications can work in closed and local environment. However it is not a good alternative for open and web-scale applications and should be avoided.

RPC does not support very well addressability. Even if it supports URI, the naming method lacks a consistent standard. Services are entirely named by developer or designer, and identifier does not refer a resource but a service contract including a group of operations. The identifier is more like a kind of distributed network handle. Most RPC-style Web services expose very few identifiers, even as few as one.

RPC adds needless complexity because in most cases, it is used for a job that plain old HTTP could handle just fine. In addition, with RPC, the HTTP protocol is reduced to a transport protocol for an enormous XML payload, which is not adapted for constrained devices. The resulting service is far too complex, impossible to debug, and will not work unless your clients have the exact same setup [52].

2.4.2 Representational State Transfer (REST)

REST can be seen as a way to help communicate the basic concepts underlying the Web. In opposition to RPC calls, REST is about working with resources instead of methods. A resource can be anything. It may be a physical object or an abstract concept. Usually a resource is something that can be stored on a computer and represented as a stream of bits. A representation is any useful information about the state of a resource [53].

In REST, service interfaces are defined according to the uniform interface constraint. In other terms all resources expose the same interfaces to the client, which enables one to reduce coupling between the client and the server. In addition, the resources are manipulated through representations and REST requires messages to be self-descriptive. The client can negotiate appropriate data formats for different types of data, such as images, text, and spreadsheets.

Addressability is a fundamental property in REST. Any interesting piece of information has a unique identifier in a format that's familiar to every user or machine. This identifier can be bookmarked, passed around between applications, and used as a stand-in for the actual resource. In addition, the statelessness constrained of REST enables greater scalability since the server does not have to maintain, update or communicate user session state. Additionally, load balancers don't have to worry about session affinity for stateless systems.

In REST, links play an important role in connecting resources. Clients can easily navigate and find their way inside the system by manipulating resources and using hypermedia as an engine of application state (HATEOAS) [53]. What is important here is that the server participate in guiding the user to the right information. This approach enables the server to evolve independently from existing clients and so reduce costs and enhance the system maintainability.

REST provides better performance than RPC because it enables one to avoid parsing and packaging SOAP messages on both sides of the client and the server, which also results in smaller payload. REST is based on existing standards widely used in web and requires no additional complexity. It recommends that the client or intermediaries should cache responses that servers mark as cacheable to eliminate some unnecessary interaction for better performance.

On one hand, we have the RPC style that adds several layers of complexity and works in closed and local environment. On the other hand, we have the REST style that follows the natural functioning of the web and that had proven successful to connect a tremendous number of heterogenous resources. So we can state that REST is more adapted for M2M and will most likely play a central role in building the future IoT.

2.5 M2M related works

In this section, we present an overview of the most relevant standard activities and research efforts aiming to move from the current siloed landscape to a more horizontal solution. For each solution, the architecture style, key features, information model, and interoperability in terms of communication and semantic are discussed. Self-management capabilities facing environment changes and support for constrained environments are presented as well. Table 2.1 shows a comparative summary of the following solutions.

2.5.1 ETSI SmartM2M

At the request of many telecom operators, ETSI initiated in January 2009 the SmartM2M [3] standard to provide a horizontal M2M service platform. SmartM2M addressed several vertical M2M domains, identified common service requirements, then derived main service capabilities required for a common M2M architecture. The main service capabilities are application enablement, generic communication, reachability, addressing, and repository, communication selection, remote entity management, security, and interworking proxy. SmartM2M offers an extensible architecture following the REST principles. However, the proposed architecture is defined following a centralized approach, which leads to serious scalability issues. The platform is designed to support different communication protocols, interwork with heterogeneous device technologies, and interface with existing device management protocols. SmartM2M standardized the communication interface, which is important for an effective interoperability. Regarding the information model, SmartM2M standardized the structure of a set of resources that can be used to interact with the platform and exchange data between applications. However, the structure of the content exchanged between applications is not standardized, which leads to semantic interoperability issues. Resources can be represented using XML or JSON content formats, and more efficient content formats (e.g. EXI) are also supported, which is useful for constrained environments but not enough. In fact, SmartM2M relies on a hierarchical naming system resulting in a large payload, which make it inefficient for constrained devices. Nothing is currently defined to provide self-management capabilities.

2.5.2 TIA TR-50

The TIA TR-50 [5] Smart Device Communications engineering committee was created in January 2009 to develop and maintain an access agnostic interface standards for bi-directional communication between devices and applications. It addresses several functional areas for M2M including reference architecture design, informational models and standard objects, protocol aspects, software aspects, conformance and testing, and security. The system architecture is formed by four functional entities called containers know as Point of Attachment, node, service, and security containers. The main services are discovery, node, application, and network management, synchronous and asynchronous communication, authentication, authorization, and accounting. TR-50 claims a REST architecture style, but in reality it looks more like an RPC style. First, it is true that the architecture enables one to send CRUD requests over HTTP. However the architecture does not make use of the fundamental REST notions of resources and links, which impedes the overall system interoperability. The proposed architecture is defined following a centralized approach, which leads to serious scalability issues. An adaptation and convergence layer is defined to enable the platform to be adapted to a given wired or wireless transport network. Applications can interact and use the framework through a well-defined API, which is agnostic to the vertical application domain. The data model enables one to handle equipment fault, preventative maintenance, energy metrics, network metrics, equipment data, and configuration management. All operations can be performed using a generic request and response representation based on the JSON format. The lack of constrained communication protocols and the absence of efficient content format make it difficult to use the platform in a constrained environment. In addition, no services were defined for semantic support and self-management.

2.5.3 OneM2M

Seven of the world's leading information and communications technology Standards Development Organizations (SDOs) launched, in July 2012, the oneM2M [6] initiative to define a globally agreed horizontal M2M service platform. OneM2M defined a common service entity to enable registration, discovery, security, and device, service, and application management, etc. OneM2M adopted a RESTful architecture. Thus all services are represented as resources. Despite the definition of new intermediate architecture entities called Middle Node (MN), the oneM2M architecture remains centralized. In fact, the role of middle nodes is limited to connect device and gateway to a centralized infrastructure node, which results in a hierarchical network. Unfortunately, it is not allowed to build a meshed network. In other terms, it is not possible to make direct communication between Application Service Node (ASN) or Application Dedicated Node (ADN), which impedes the system scalability. The platform is designed to support different communication protocols, interwork with heterogeneous device technologies, and interface with existing device management protocol. OneM2M standardized the communication interface, which is important for an effective interoperability. Regarding the information model, oneM2M standardized the structure of a set of resources that can be used to interact with the platform and exchange data between applications. However, the structure of the content exchanged between applications is not standardized, which leads to interoperability issues. OneM2M support both hierarchical and non hierarchical resource structure, which offer more flexibility to support constrained devices. However, the collection pattern is not considered in the OneM2M resource architecture. Lack of collection resources results in large payload and additional URI parameters, which make it difficult to use the platform in constrained environments. Some works are in progress to define a core ontology for OneM2M to support semantic interoperability between applications and devices. There is no support for self-management to decrease the system complexity.

2.5.4 OMA Service Enablers

The Open Mobile Alliance (OMA) published a set of service enablers [7] to help in the development, deployment and operation of applications and services in both fixed and mobile environments. OMA Device Management (DM) is one of the most important enablers addressing the need for management of mobile devices such as mobile phones, PDAs, and tablet computers. It is intended to support device provisioning, configuration, Software Upgrades, and fault management. However, OMA DM do not address the issues from technical fragmentation and was not adapted to constrained M2M devices. To overcome these challenges, the OMA Lightweight M2M (LWM2M) enabler was defined to provide a light and compact secure communication interface along with an efficient data model, which together enable device management and service enablement over constrained sensor networks. LWM2M is designed to be extensible to meet the requirements of vertical applications. It offers a RESTful architecture and utilizes the CoAP protocol as the underlying transfer protocol over UDP and SMS bearers. Even though LWM2M proposes an efficient abstraction device model, it does not support semantic interoperability across M2M devices and applications. It does not propose solutions for self-management as well.

2.5.5 IETF ROLL & CORE

The Internet Engineering Task Force (IETF) is very active in realizing the IoT especially through the Routing Over Low power and Lossy networks (ROLL) and the Constrained RESTful Environ-

ments (CORE) working groups [8]. IETF ROLL is addressing routing and path selection issues for the IPv6 protocol tailored to Low power and Lossy network (LLN) and resource constrained devices. It takes into consideration various aspects including reliability, connectivity, security and manageability while permitting low-power operation with very low memory and CPU pressure in networks potentially comprising a very large number of nodes. IETF CORE developed a framework for resource-oriented applications intended to run over the IP protocol on constrained environments. CORE defined the Constrained Application Protocol (CoAP) as a web transfer protocol for use with constrained networks to connect billions of node in the Internet of Things. This includes applications to monitor simple sensors, to control actuators, and to manage devices. CoAP enables synchronous communication to create, read, update and delete a resource on a device. It enables asynchronous communication to subscribe to a resource on a device and receive notifications. Non-reliable multicast messages are supported for device group managing. Even though CoAP supports built-in discovery of services and resources, it lacks semantic support. Self-management is not address by CORE.

2.5.6 IoT-A

The IoT-A [9] solution proposes an architectural reference model for the interoperability of IoT systems. It defines an initial set of key building blocks to enable seamless end-to-end communication between IoT devices. The reference architecture includes a domain model that describes the required generic components, an information model explaining the semantic data, and a communication model for interconnecting heterogeneous devices. A set of ontologies for entities, resources, devices and services based on the SSN and OWL-S ontologies are defined. Adaptive mechanisms are specified for distributed orchestration of IoT resource interactions exposing self-management properties in order to deal with the complex dynamics of real world environments. However, IoT-A does not provide any specific optimization for constrained environment.

2.5.7 OpenIoT

The OpenIoT [10] platform combines internet-connected objects, cloud computing, semantic, and self-management techniques to facilitate the use of sensors in vertical applications domains. OpenIoT offers RESTful web services accessible via HTTP protocol. A sensor middleware is defined to collect and process data from heterogeneous sources, including physical devices, sensor processing algorithms, and social media processing algorithms. A Cloud Computing Infrastructure is used to support the storage of data along with their associated metadata information in a scalable and elastic manner. A Semantic Directory Service is defined to support registration management for sensors and services. The annotation of the registration profile with metadata compliant to the W3C Semantic Sensor Network (SSN) ontology enables the semantic search and discovery of sensors and services. An autonomic service control loop is defined to provide self-management capability to make the infrastructure more reactive by means of data and service control. It enables dynamic deployment of applications and services to enhance system system performance, reliability and scalability. The use of OpenIoT in constrained environments is not well addressed.

2.5.8 BETaaS

The BETaaS [11] platform, which refers to Building the Environment for the Things as a Service, aims to provide a common cloud-based reference model offering service capabilities to enable inter-

M2M Platform	Architecture Style	Organizational Structure	Communication Protocol	Constrained Environment	Self-Management	Semantic Support
SmartM2M	REST	Centralized	HTTP, CoAP	Partial	No	No
TIA TR-50	RPC	Centralized	HTTP	Partial	No	No
OneM2M	REST	Centralized	HTTP, CoAP, MQTT	Partial	No	Yes
OMA LWM2M	REST	Centralized	CoAP	Yes	No	No
IETF CORE	REST	Decentralized	CoAP	Yes	No	No
IoT-A	RPC	Centralized	HTTP	No	Yes	Yes
OpenIoT	REST	Centralized	HTTP	No	Yes	Yes
BETaaS	REST	Decentralized	HTTP, CoAP	Yes	Partial	Yes

Table 2.1: Comparative table of interoperable M2M solutions

operability, scalability, fast deployment. It consists of building a service platform for the IoT over a meshed overlay of gateways that seamlessly integrate existing heterogeneous M2M devices. It relies on a decentralized architecture accessible via a RESTful API, which is efficient for large scale deployment and big data management. In order to unify the information that comes from heterogeneous resources and applications, BETaaS defines a semantic model based on existing ontologies like SSN. Such model enables one to infer information not explicitly reflected in the ontologies. BETaaS claims that the proposed ontology enables one to support context awareness and so reduce the complexity of the IoT system. However, no concrete self-management mechanism was defined to clearly overcome the increasing complexity challenge. A CoAP Adaptation plugin is provided to support CoAP-enabled sensors in constrained environment.

2.6 SmartM2M service architecture

In this section, we focus more on the SmartM2M standard given that it is the most mature horizontal solution. The extensible architecture of SmartM2M enables us to integrate additional services to meet M2M challenges. We illustrate the SmartM2M high level system architecture, the service capability framework, the resource structure, and the interface procedure.

2.6.1 High level system architecture

A considerable effort is done by ETSI in defining the SmartM2M specifications [54–56]. The proposed solution offers a RESTful Service Capability Layer (SCL) accessible via open interfaces to enable developing services and applications independently of the underlying network.

As illustrated in Figure 2.1, the SmartM2M high-level system architecture includes the concept of the M2M device domain, as well as the network and application domains. The M2M device domain is composed of the following elements:

- M2M Device: a machine that runs an M2M device SCL (DSCL) that can be queried by M2M device applications (DAs) via an open interface. An M2M device can connect directly to the M2M Network or indirectly via a gateway that acts as a network proxy.

- M2M Area Network: provides connectivity between M2M devices, and also between the devices and the M2M gateways. It is based on existing technologies such as Zigbee, Phidgets, M-BUS, KNX, etc.

- M2M Gateway: a machine that runs an M2M Gateway SCL (GSCL) that can be queried by M2M Gateway Applications (GAs) or DAs via an open interface. It identifies and addresses heterogeneous devices, and interconnects M2M area networks with different protocol technologies by performing mapping/converting operations.

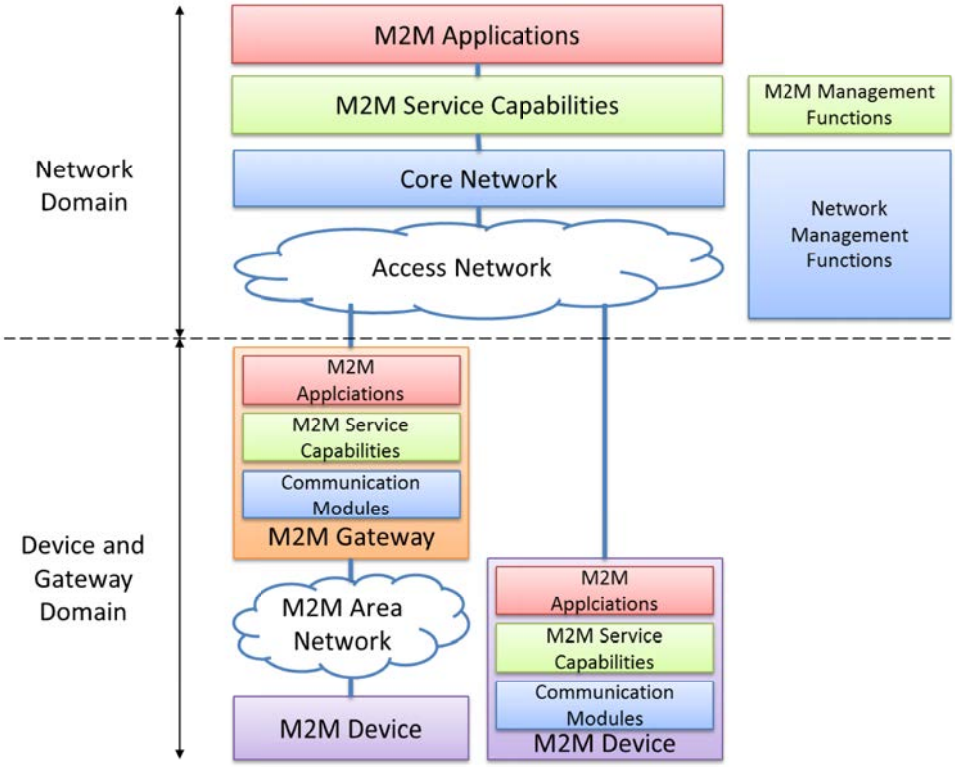

Figure 2.1: SmartM2M High level architecture

The network and application domains are composed of the following elements:

- Wide Area Network provides the core network with access networks to enable M2M devices and gateways to communicate with the M2M network. It is based on existing technologies such as xDSL, GERAN, UTRAN, eUTRAN, W-LAN, WiMAX, etc.
- M2M Network provides an M2M Network SCL (NSCL) that can be queried by M2M Network applications (NAs) via an open interface.
- M2M Service Capabilities Layer (SCL) provides a horizontal service layer for M2M applications abstracting the complexity of heterogeneous devices and enhancing M2M interoperability.
- M2M application provides a domain-specific software application that interacts with the M2M machines by querying the service layer through a set of open interfaces.

2.6.2 Service capability framework

Figure 2.2 provides the framework that is used to build the SmartM2M architecture. It shows a service layer architecture that is articulated around a set of SCs, either in the network and application domains or in the device domaine:

- Application enablement(AE) provides a single point of contact to M2M applications using XML files and HTTP protocol but can be mapped easily to other protocol technologies such as CoAP.
- Generic communication(GC) provides a single point of contact for communication between different SCLs using XML files and HTTP protocol but can also bind to other protocols. It makes use of existing network connectivity and manages all security aspects for secure session establishment and tear down.
- Reachability, addressing, and repository (RAR) provides persistence capability to store M2M resource states within the SCL. It enables one to Create, Retrieve, Update and Delete (CRUD) M2M resources, to store information related to M2M applications and SCls registrations, and to manage data required for publish/subscribe communication.
- Communication selection(CS) provides network selection algorithms and mechanisms based on policies for M2M machine with multiple interfaces and bearer such as Ethernet, Wifi, GPRS or 3G.
- Remote entity management(REM) interworks with existing REM standards such as the OMA-DM by performing mapping/converting operations. It provides functions pertaining to device/gateway life cycle management, such as software and firmware upgrade and provides mechanisms for fault and performance management.
- SECurity(SEC) implements bootstrapping, authentication, authorization, and key management functions to establish secure communication between M2M applications and SCLs based on the TLS-PSK protocol.
- Interworking proxy(IP) allows non-ETSI-compliant devices such as Zigbee, Phidgets, and many other technologies to interwork with the SmartM2M standard by performing mapping/converting operations. In the smart building domain, a specific application payload is proposed based on the OBIX information model to enhance interoperability.

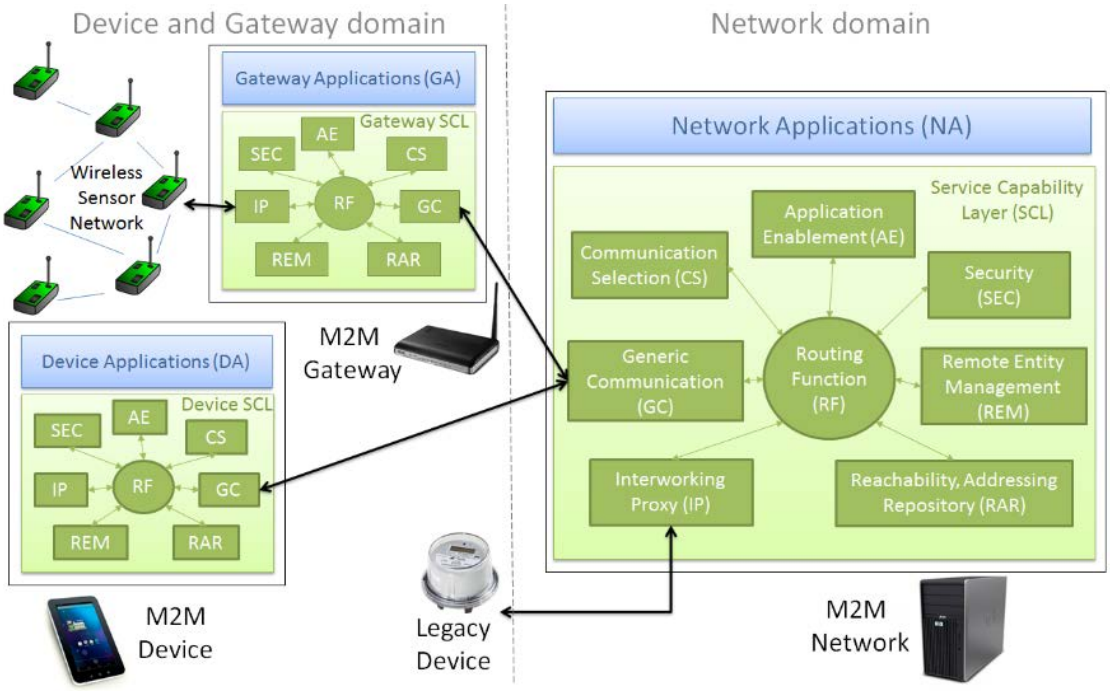

Figure 2.2: SmartM2M functional architecture

Three reference points based on open APIs are specified:

- mIa reference point: allows a NA to communicate remotely with the NSCL
- dIa reference point: allows a DA to communicate with the DSCL or the GSCL where it is residing. It allows also a GA to communicate with the GSCL where it is residing
- mId reference point: allows DSCL and GSCL to communicate with the NSCL on the service level and vice versa. The mId uses core network connectivity functions as an underlying transport

2.6.3 Resource structure

SmartM2M standardized the M2M SCL resource tree. Each SCL resource is referenced with a global identifier and has a type, a set of attributes, a set of methods that operate on it, and some links to other resources. Most important SCL resources are listed below:

- sclBase resource describes the hosting SCL, and is the root for all other resources within the hosting SCL.
- scl resource stores information related to M2M SCLs residing on other M2M machines after successful mutual authentication. It enables SCLs interactions using retargeting operations.

- application resource stores information about the application after a successful registration on the hosting SCL.
- container resource acts as a mediator for data buffering to enable data exchange between applications and SCLs.
- accessRight resource manages permissions and permission holders to limit and protect the access to the resource tree structure.
- group resource enhances resources tree operations and simplifying the interactions on the API interfaces by adding the grouping feature. It enables an issuer to send one request to a set of receivers instead of sending requests one by one.
- subscription resource stores information related to subscriptions for some resources. It allows subscribers to receive asynchronous notification when an event happens such as the reception of new sensor event or the creation, update, or delete of a resource.
- collection resource groups common resources together. It is used at different places such as SCLs, applications, or containers.
- announced resource contains a partial representation of a resource in a remote SCL to simplify discovery request on distributed SCLs.
- discovery resource acts as a search engine for resources. It enables one to retrieve the set of resources URIs matching a specific filter criteria and constraints.

2.6.4 Interface procedures

Interface procedures define the set of primitives and interactions on the mIa, dIa, and mId interfaces that allow resource manipulation. Ultimately, the objective of the interface procedures is to allow exchange of data between applications within the M2M system. However, data exchange can only take place when a certain number steps have been accomplished:

- SCL discovery defines the procedures that allow a DSCL or GSCL to discover an NSCL so as to perform registration.
- SCL registration defines the procedures that allow a GSCL or a DSCL to register to a NSCL once discovery has been performed. SCL registration is a necessary procedure to allow the SCL to start resource management procedures. SCL registration assumes that SCLs have performed appropriate authentication and authorization mechanisms.
- Applications registration defines the set of procedures that allows an application (NA, DA, or GA) to register to its local SCL. Application registration is the necessary step for an application to become known and to start exchanging data using the resource-based procedures.
- Access-right management consists of manipulating (creation, deletion, update, retrieve) resources pertaining to access rights.
- Container management consists of procedures that allow the exchange of application data through the use of specific resources, referred to as container resources.

- Group management consists of manipulating groups of resources. Group resources allow for smoother interaction between applications and SCLs and among SCLs.
- Resource discovery allows the discovery of resources stored on a specific SCL or resources announced on that SCL through the use of filter criteria. The filter criteria may consist of a combination of attributes such as the creationTime and searchString.
- Collection management defines the set of procedures to manage collection resources.
- Subscription management defines the set of procedures allowing an application or an SCL to subscribe and be notified when specific subscription criteria are matched.
- Resource announce/de-announce defines the set of procedure to allow resources to be announced and de-announced towards a remote SCL.

2.7 OM2M standard-based service platform for M2M interoperability

Based on the SmartM2M standard, we designed and developed the OM2M [13] platform that provides an extensible and flexible operational architecture for M2M cross-domain interoperability. OM2M system architecture, service dynamic discovery, and main plugins are explained using UML diagrams and screenshots.

2.7.1 OM2M platform overview

The OM2M architecture is composed of users, business applications, servers, gateways and devices. After a mutual authentication, a server and a gateway can exchange data using simple REST requests through the "mId" interface. The communication through the "mId" interface can be done via HTTP or CoAP depending on specific policies. A gateway enables heterogeneous and constrained devices to integrate the platform through the "dIa" interface or using an interworking proxy. On one hand, connected devices expose their services to business applications in a standardized way simply by creating resources via the gateway. On the other hand, business applications register to the server and use the "mIa" interface to manage the connected devices simply by dealing with created resources. OM2M enables registered applications to discover, monitor and control heterogeneous devices in a seamless way by hiding the complexity of the specific technologies used by devices behind the gateway. Figure 2.3 shows an example of a typical M2M architecture composed of a server and a gateway. Using OM2M, end user devices, data analytic servers, and SCADA interfaces are able to manage heterogeneous devices having MQTT, Zigbee, 6LowPAN, KNX, PHIDGETS or any other protocol without the need to learn such vendor-specific technologies.

2.7.2 SCL Resource tree

OM2M enables seamless integration of heterogeneous devices by providing services via open interfaces and exposing data via a generic resource tree. The resource tree enables one to efficiently structure relevant data related to an M2M machine. Each M2M machine has an "SclBase" resource that can contain a set of "Application" resources. The "Application" resource can be created by any

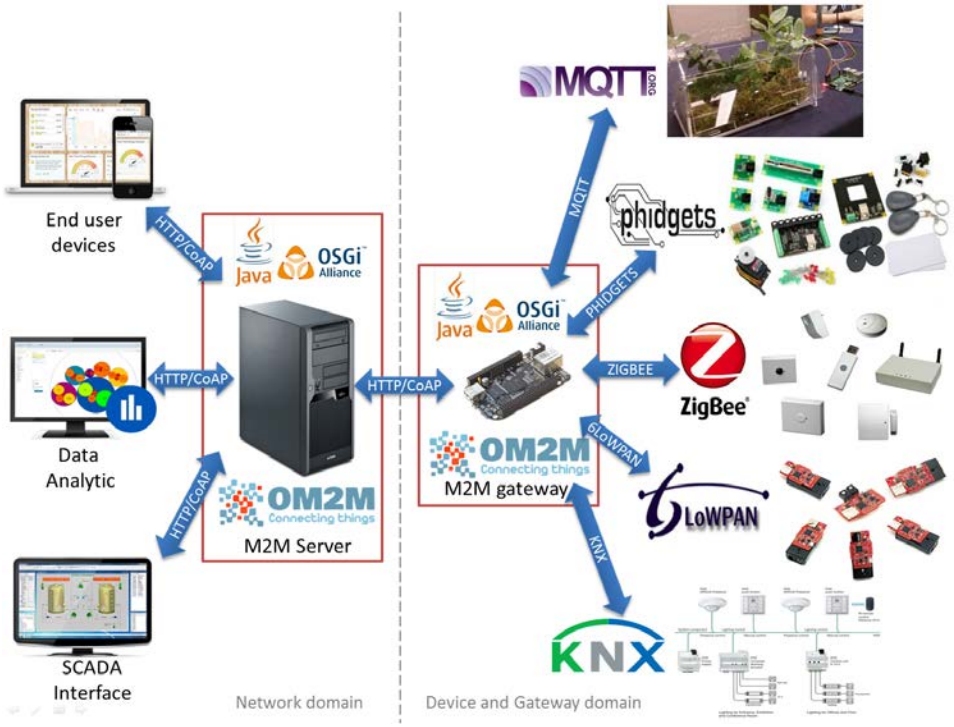

Figure 2.3: OM2M high level architecture

business application or any physical device connected to the M2M device. It may contain several "container" resources to structure its data. A "Container" resource may contain several "contentInstance" resources that serve as black boxes to store the exchanged data. Other types of resources are used to handle access right authorization, group broadcast, subscription and notification, etc. Figure 2.4 illustrates a concrete example including different type of devices and making use of the described resources. In this example, the user's house is equipped with a smart meter connected to a local gateway, which is connected to a remote M2M server. The end user wants to access his home electric energy consumption remotely using his Smartphone anytime anywhere.

Once started, the smart meter creates an "Application" resource called "SMART_METER_1" on the gateway resource tree. In this application, the smart meter creates two "Container" sub-resources. The first container is called "DESCRIPTOR" and is dedicated to store the smart meter meta-data such as manufacturer, location, etc. The second container is called "DATA" and is dedicated to store the measured data. In the first container, the smart meter creates one "contentInstance" resource containing the smart meter description. In the second container the smart meter periodically creates "contentInstance" resources containing the consumed energy. The end user can use his smartphone to communicate with the server, which acts as a proxy to retarget the user requests to gateway. He can read the description from the "DESCRIPTOR" container to discover the smart meter description. Then, he can retrieve the measured data from the "DATA" container to monitor the energy consumption. He can also subscribe to this container to receive the measured data asynchronously using notification. OM2M acts as an intermediate service platform

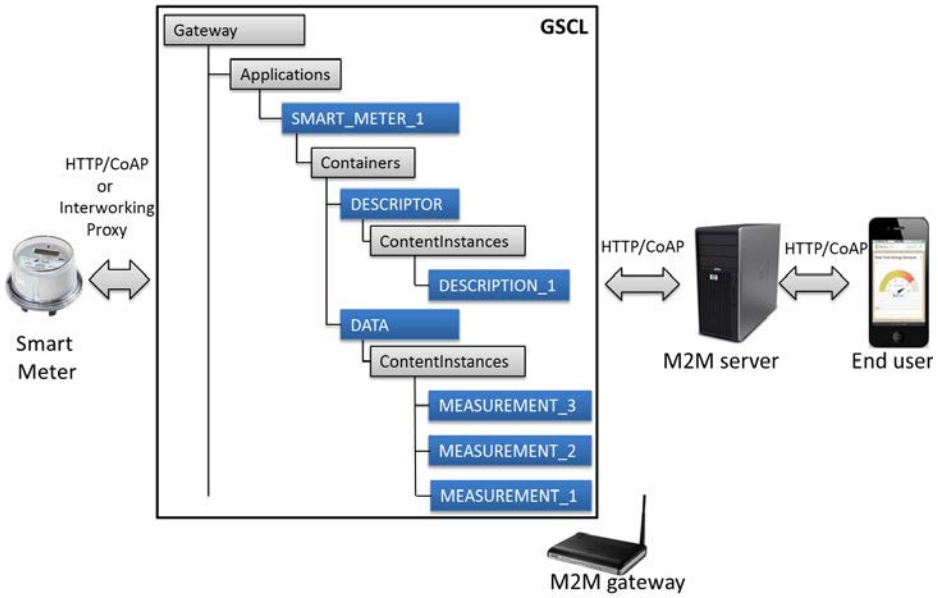

Figure 2.4: OM2M resource tree example

that decouples users and business applications from vendor-specific technologies.

2.7.3 Main building blocks

OM2M offers a modular architecture running on top of an OSGi runtime that can be deployed in an M2M server, a gateway, or a device. Each SCL is composed of small tightly coupled plugins, each one offering specific functionalities. A plugin can be remotely installed, started, stopped, updated, and uninstalled without requiring a reboot. It can also dynamically discover the addition or the removal of services via the service registry and adapt accordingly. OM2M building blocks are illustrated in Figure 2.5.

The CORE plugin is the main module that shall be deployed in each SCL. It provides a protocol independent service for handling REST requests. Two communication binding plugins are added to support HTTP and CoAP. A remote entity management plugin is provided to enable device firmware updates based on the OMA-DM protocols. Various interworking proxy plugins are added to enable seamless communication with legacy technologies such as Zigbee, KNX, MQTT, 6LowPan, and Phidgets, or XMPP [57]. The TLS-PSK protocol is used to secure M2M communications based on pre-shared keys. Two experimental plugins are designed and integrated to enhance OM2M capabilities. The first one automates resource discovery and self-configuration based on the Autonomic Computing (AC) paradigm. The second one enhances the platform scalability by optimizing data routing based on a promising Information Centric Networking (ICN) approach.

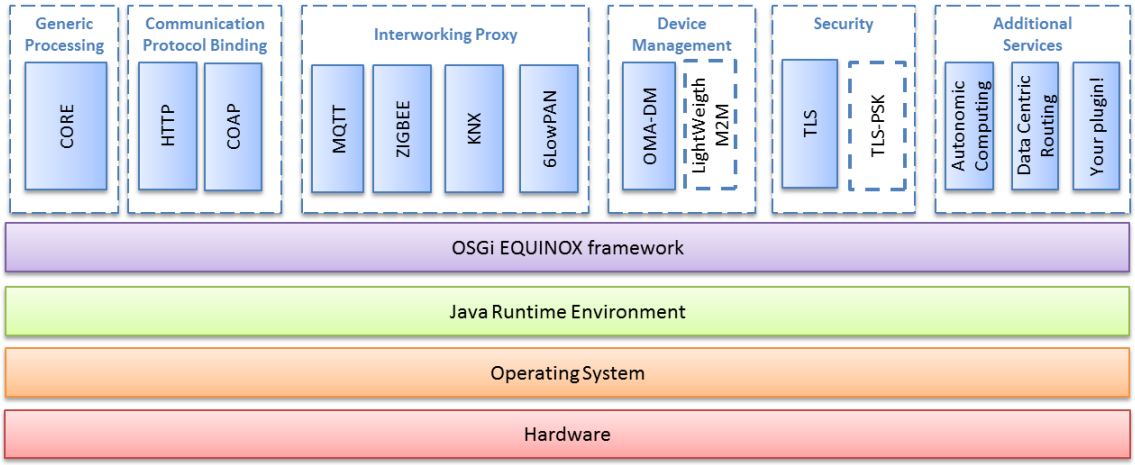

Figure 2.5: OM2M building blocks

2.7.4 Components diagram

Figure 2.6 illustrates three main components of OM2M including the core, interworking proxies and communication binding modules. The CORE module implements the "SCL Service" interface to handle standardized REST requests. It receives a protocol-independent REST request and returns protocol-independent REST response. The Router defines a single route to handle every request in a resource controller component simply by analyzing the received request URI and method. The "Controller" component implements CRUDE (Create, Retrieve, Update, Delete, and Execute) methods for each resource. It performs required checking operations such as access right authorization, and resource syntax verification. The "Data Access Object" component provides an abstract interface to encapsulate all access to resource persistent storage without exposing details of the database. It interfaces with external database drivers to support different database systems. The "Notifier" sends notifications to all interested subscribers when a resource is created, updated or deleted. It performs filtering operations to discard events not of interest to a subscriber. The "Announcer" announces a resource to a remote SCL to make it more visible and accessible to other machines. It also handles resource de-announcement. The "Client Selector" component manages discovered Communicaton Bindings (CBs) clients implementing the "CB Service". It acts as a proxy to send a generic request via the appropriate communication protocol. The "Interworking Proxy Selector" component manages discovered Interworking Proxy Units (IPUs) implementing the "IP Service" interface and acts as a proxy to call the correct IPU controller. The "Device Manager Selector" component handles discovered Remote Entity Managers (REMs) implementing the "DM Service" interface, and acts as a proxy to call the appropriate Device Manager Unit controller.

Specific protocol communication bindings can be added to extend the CORE plugin capabilities. Each binding plugin is composed of three main components that enable one to bind to a particular communication protocol. The "Specific Server" component exposes the API on a specific application protocol and content format. It is responsible for receiving requests from external specific entities, calling the CORE plugin, and returning-back the response to the specific entities. The "Specific Client" component enables one to send requests according to a specific application protocol and

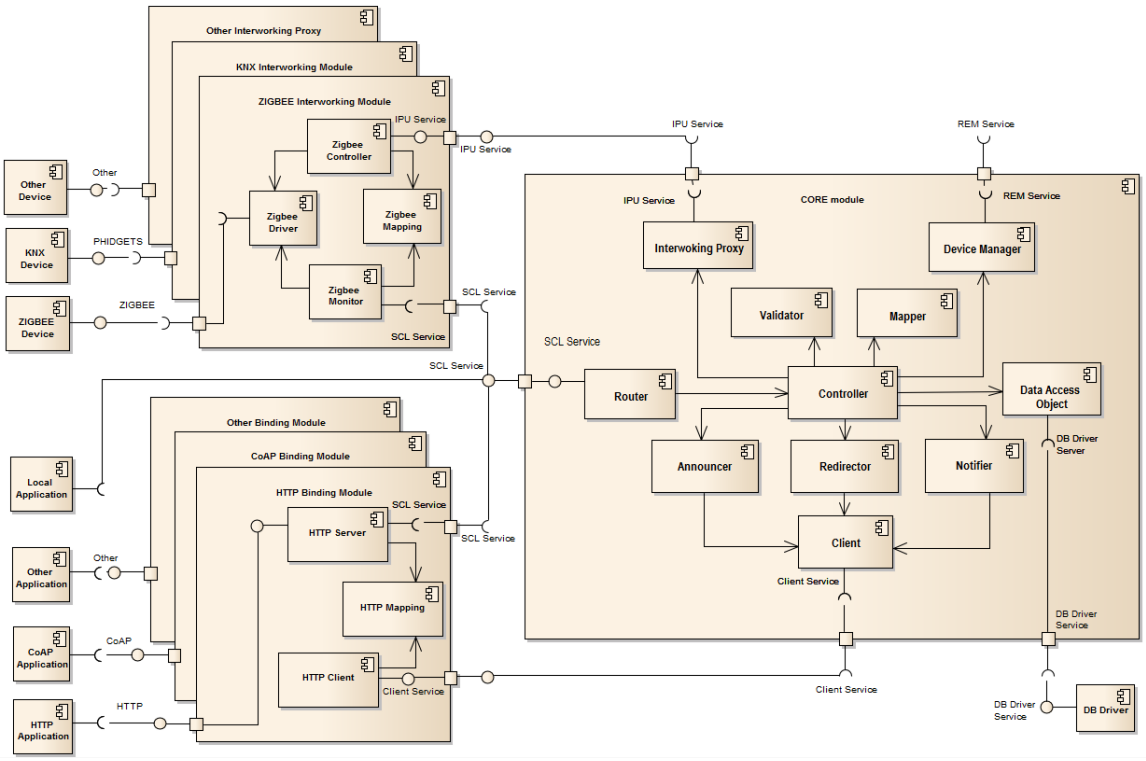

Figure 2.6: OM2M components diagram

content format. It is responsible of receiving requests from the CORE plugin, sending them to external specific entities, and returning-back the response to the CORE plugin. The "Specific Mapping" component enables one to convert data from a protocol-dependent format to a protocol-independent format and vice versa. A HTTP and a CoAP communication binding plugins are designed and integrated to OM2M. Others binding modules can be added using the same approach.

Specific interworking proxy plugins can be added to extend the CORE module capabilities to interwork with vendor-specific devices. Each interworking proxy plugin is composed of four components and enables one to interwork with a particular type of device. The "Specific Monitor" component enables one to discover vendor-specific devices and collect relevant data. It is responsible of collecting data from legacy devices and sends it to the CORE plugin. The specific "Controller" component enables one to send request to specific devices. It is responsible of receiving data from the CORE plugin and sending it to specific devices. The specific mapping component offers required tools to convert data from a vendor-specific format to a generic format and vice versa. The "Specific Driver" component enables one to access hardware functions without needing to know precise details of the hardware being used. A ZIGBEE, PHIDGETS, KNX interworking proxy plugins are designed and integrated to the platform. Others interworking proxy plugins can be added using the same approach.

2.7.5 Dynamic discovery of services

OM2M offers a modular architecture extensible via plugins. Figure 2.7 depicts how the plugin puzzles are discovered and connected together. On one hand, the CORE plugin registers a single "SCL Service" interface to enable newly deployed plugins to discover the CORE plugin, access the "Router" component, and use it through local library calls. On the other hand, the CORE plugin listens permanently to three other types of services, which are "CB Service", "IP Service", and "DM Service" to discover respectively new Communication Bindings, Interworking Proxies, and Device Management plugins. The "Plugin Manager" handles three bidirectional tables each one is dedicated to a specific extension type. After a successful deployment and initialization, each plugin registers its service interface to notify the CORE plugin. Once notified, the "Plugin Manager" gets the plugin implementation, and adds it to the corresponding table using a well-defined identifier. If a plugin is stopped or uninstalled, the "Plugin Manager" removes it from the corresponding table. For each table, a "Selector" component is defined to select the best plugins to satisfy received requests.

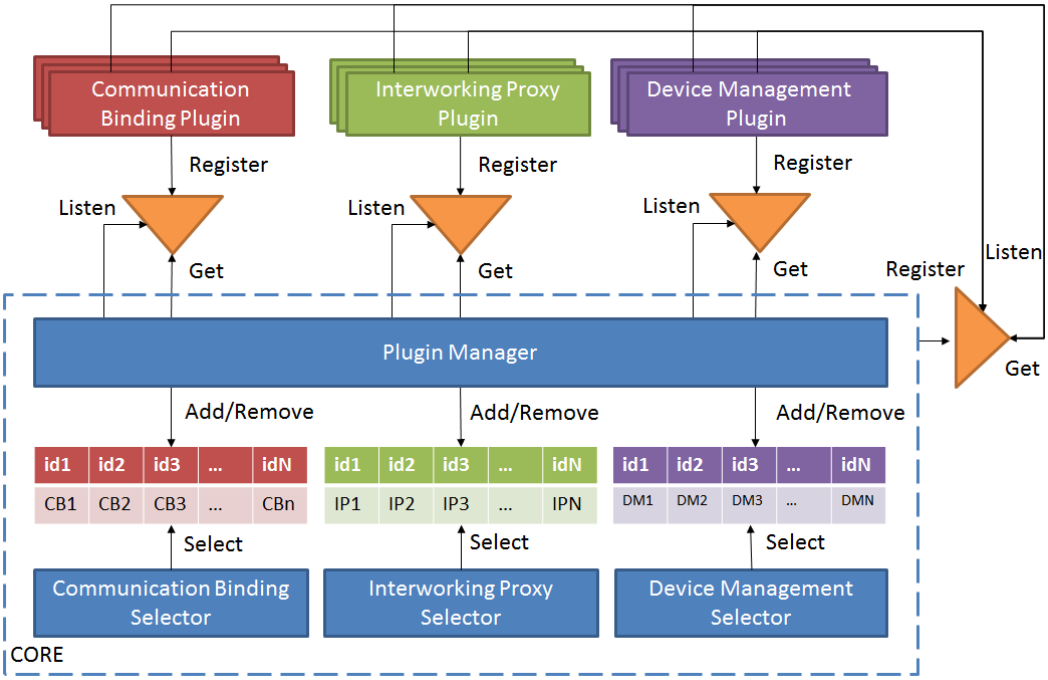

Figure 2.7: OM2M plugins discovery

2.7.6 Protocol-independent CORE module

OM2M relies on a protocol-independent CORE module extensible via plugins. The "CORE Service" interface is described in Figure 2.8. It offers a single method called "doRequest" to interact with the CORE plugin. Local applications and additional plugins can use the "doRequest" method to

manage any type of resources. A generic data structure is defined to represent the request and the response according to the REST style. The "RequestIndication" class describes all required information to perform a generic REST request. It contains mainly the following attributes:

- Method: contains a verb describing how to process the received request. It can be "CREATE", "RETRIEVE", "UPDATE", "DELETE", or "EXECUTE".
- Base: defines the protocol-specific part of the URL. In general, it contains the protocol schema, the IP address, the port, and the context. E.g. "http://192.168.0.3:8080/om2m".
- Target: defines the protocol-independent part of the URL. It targets the resource involved in the request without including the query string parameters. E.g "sclBaseId/applications/appId".
- Representation: contains a format-independent representation of the resource involved in the request. This attribute is used only for creation and update requests.
- From: contains the identity of the requesting entity required for resource authorization. It can be defined as a token including a user name and a password. E.g "name:password".

The "ResponseConfirm" class describes all needed information to return a generic REST response. It contains mainly the following attributes:

- StatusCode: defines a status describing the result of the request. It can define a successful status such as "STATUS_OK", "STATUS_CREATED", etc. It can define also an error status such as "STATUS_NO_FOUND", "STATUS_SERVICE_UNAVAILABLE", etc.
- Representation: contains a format-independent representation of the returned resource. For a successful request, it can contain the retrieved, created, or updated resource in the case of, respectively, a retrieve, creation, or update request. For an unsuccessful request, it can contain a generic representation giving more details about the error message.
- Location: contains the protocol-independent location of the returned resource, mainly used in creation requests.

The sequence diagram of the CORE plugin illustrated in Figure 2.9 depicts the components involved in a resource creation with the sequence of messages exchanged to carry out the scenario, and shows how processes operate with one another and in what order. In this scenario, the issuer is a local application that interacts with the CORE plugin using a direct library call. It builds a protocol-independent request of resource creation using the "RequestIndication" object then calls the "doRequest" method provided by the "CORE service". The "Router" receives the call and finds the route for the request based on the URI and the method name. If the target SCL is different of the sclBase, the "Router" routes the request directly to the "Redirector", which find the appropriate remote SCL. The "Redirector" then calls the "CB Service" to send the request to the remote SCL using a specific communication protocol. In this case the request is processed remotely and the resource is created on the remote SCL. The response is sent back from the remote SCL to the hosting SCL to the local application. If the target SCL is equal to the "SclBase", the "Router" routes the request to the corresponding "Resource Controller". The "Resource Controller" checks the issuer access right and validates the resource representation, then calls the "DAO" to

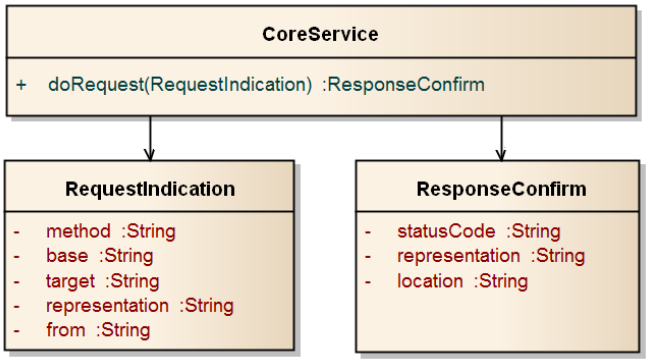

Figure 2.8: Protocol-independent exchange structure

store the resource on the database. It builds a "ResponseConfirm" object and returns it to the local application. The "Resource Controller" calls the "Notifier" to notify the interested subscribers. The "Notifier" builds a protocol independent request for each subscriber then calls the "CB service" in a separate thread to send the notification request using the appropriate communication protocol. If needed, the "Resource Controller" calls the "Announcer" to create announcement in remote SCLs. The "Announcer" builds a protocol independent request for each remote SCL, then calls the "CB Service" in a separate thread to send the announcement request using the appropriate communication protocol.

2.7.7 Binding to multiple communication protocols

Since the CORE plugin is accessible via a simple local API and is designed to handle protocol-independent requests, it becomes easy to extend its communication capabilities through specific protocol bindings simply using a bidirectional mapping. Figure 2.10 shows a concrete example of an HTTP mapping of an application creation request. It explains how the HTTP request header and body fields are mapped to the generic "RequestIndication" attributes, and how the generated "ResponseConfirm" attributes are used to build the HTTP Response. This example shows the first direction of the mapping mechanism, in particular when the SCL is acting as a server. However, the SCL can also act as a client to perform retargeting, notification, or announcement operations. This is why a mapping in the other direction is required. So that an HTTP request can be converted from the generated "RequestIndication" object, and also a "ResponseConfirm" object can be converted from the returned HTTP response.

Using the same approach, different protocol binding modules can be added to extend the CORE communication capabilities. Figure 2.11 shows a sequence diagram describing the components involved by the integration of the HTTP and CoAP communication bindings modules. Only one SCL is considered to simplify the message flow of the sequence diagram, but it is possible to consider a distributed example including many SCLs exchanging data via the retargeting mechanism. In the current example, a remote HTTP application, a CoAP sensor and a CoAP actuator communicate together in a seamless way through the SCL simply by managing resources via REST requests. Once the CoAP sensor measures a new event, it sends a CoAP request to the "CoAP server" to create a "contentInstance" resource on the SCL, which converts the request to a generic format, then

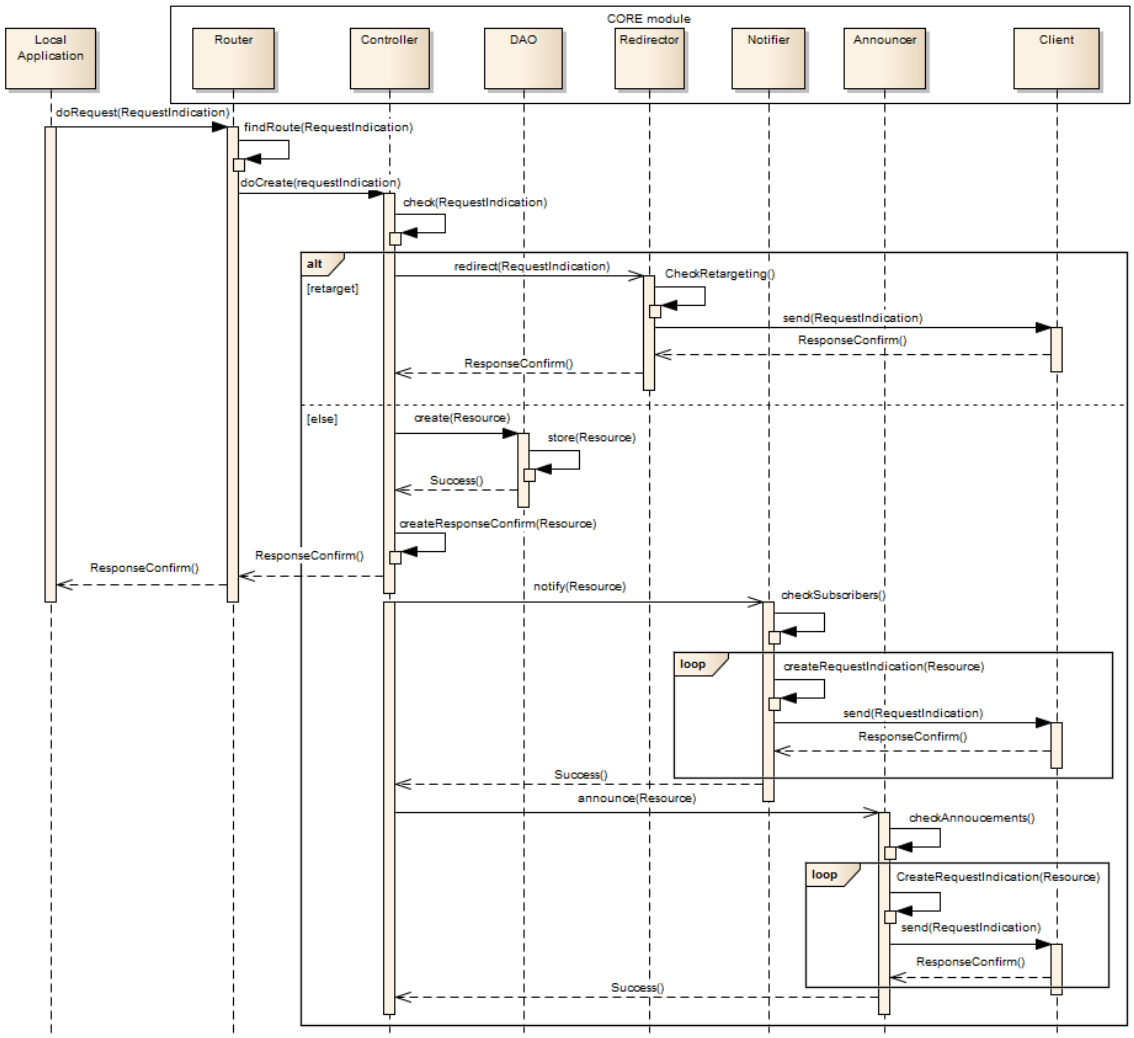

Figure 2.9: Protocol-independent CORE module

sends it to the CORE plugin. The CORE plugin checks the request, stores the “contentInstance” resource, then generates a generic response which is converted to a CoAP response, then returned back to the CoAP sensor. The CORE plugin checks the subscribers list, generates a generic request, and sends it to the HTTP client to notify the remote HTTP application. The “HTTP client” calls the “HTTP mapping” to convert the generic request to HTTP request, then sends it to the HTTP application, which returns back an HTTP response. The “HTTP client” calls the “HTTP mapping” to convert the specific request to a generic response, and returns it back to the CORE plugin.

Let’s suppose that, after analyzing the received event, the HTTP application decides to change the state of the CoAP actuator. It sends an HTTP request to the “HTTP server”, which converts the request to a generic request using the “HTTP mapping”, then sends it to the CORE plugin.

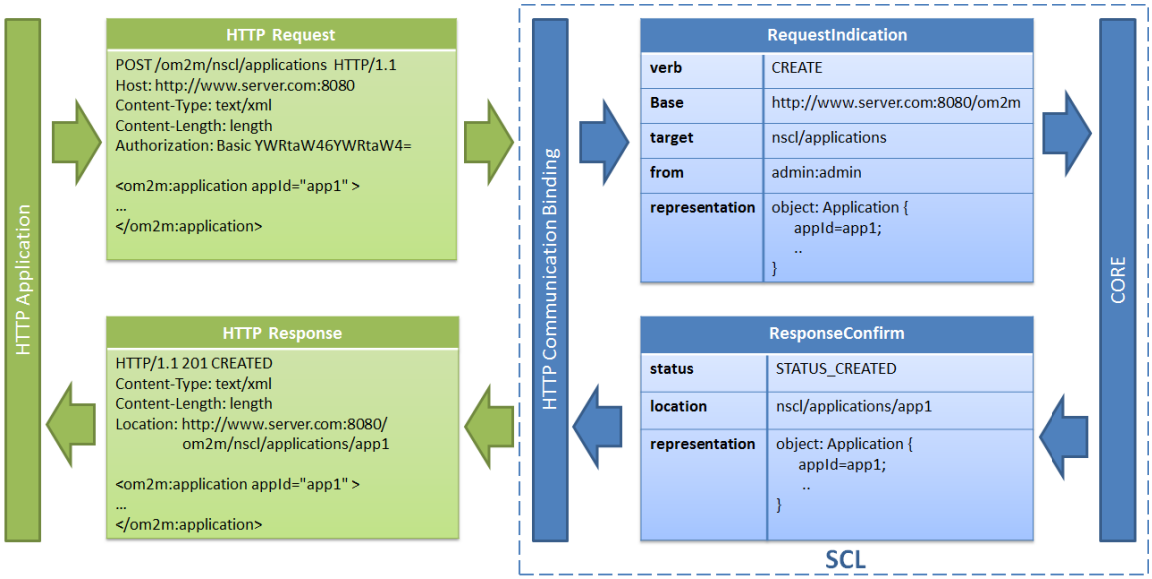

Figure 2.10: HTTP binding example

The CORE plugin retargets the generic request to the CoAP actuator by calling the "CoAP client". The CoAP client makes the conversion using the "CoAP Mapping", then sends the CoAP request to the "CoAP Actuator". The "CoAP Actuator" executes the request, then returns a CoAP response, which is converted to a generic response and returned to the CORE plugin. The CORE plugin gives the generic response to the HTTP client, which converts it to an HTTP response, then returns it back to the "HTTP Application".

2.7.8 Interworking with legacy devices

The CORE plugin can be extended to interwork with legacy devices by the means of interworking proxy plugins. Figure 2.12 presents a sequence diagram explaining how a zigbee interworking proxy can be added to the CORE plugin, and how its components are linked together to seamlessly integrate ZIGBEE devices to the platform. In this example, only one SCL is considered to simplify the exchanged message, but it is also possible to consider a distributed scenario where many SCLs interact via the retargeting mechanism. Once the "Zigbee device" is connected, it sends a zigbee hello message to the coordinator, which sends the received message to the "Zigbee monitor" via the "Zigbee Driver" using USB. The "Zigbee Monitor" calls the "Zigbee Mapping" to build a corresponding generic request for the creation of an application, then sends it to the CORE plugin, which store the application resource on the SCL and returns back a generic response. The CORE plugin checks the subscribers list, generates a generic request for the notification, and gives it to the "HTTP binding plugin", which sends an HTTP request to the "HTTP application". The "HTTP Application" checks the request, and sends back an HTTP response to the "HTTP binding plugin", which returns it back to the CORE plugin in a generic format.

Let's suppose now that, after analyzing the received notification, the HTTP application is

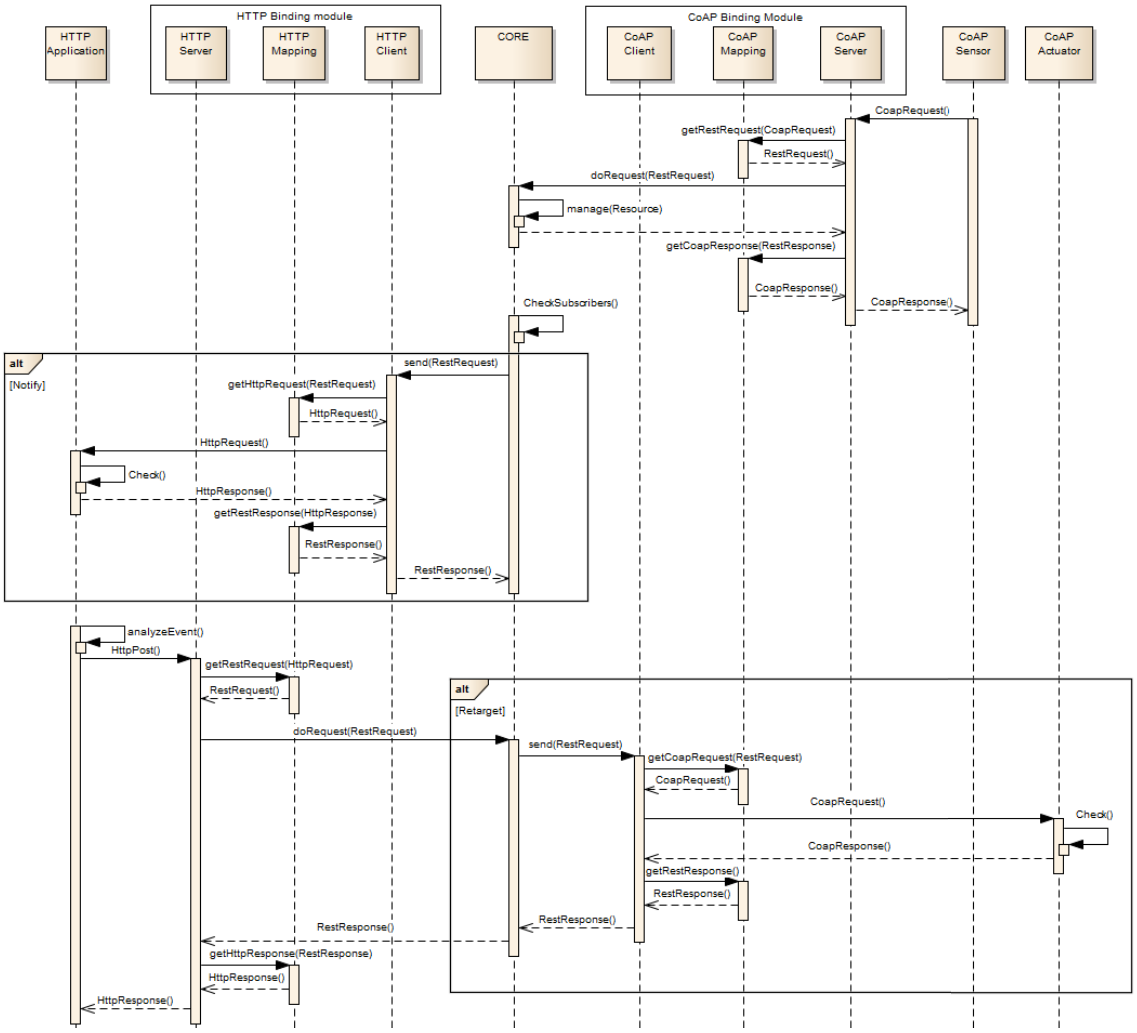

Figure 2.11: Binding to multiple communication protocols

interested of retrieving a specific attribute from the new zigbee device. It sends an HTTP request to the “HTTP binding plugin”, which makes it generic then sends it to the CORE plugin. The CORE plugin finds the zigbee interworking proxy and calls the “Zigbee Controller”. The “Zigbee Controller” uses the “Zigbee Mapping” to generate the Zigbee message from the received generic request, then sends it to the “ZigbeeDriver”. The “Zigbee Driver” sends the messages via USB to the Zigbee Coordinator that sends it to the “Zigbee device” using the Zigbee protocol. The “Zigbee Device” checks the message, then returns the zigbee response, which is converted to a generic response and returned to the CORE plugin. The CORE plugin returns the generic response to the “HTTP binding plugin”, which convert it to HTTP and returns it back to the “HTTP Application”.

In theory, the information defining objects and their relations can be fully structured using

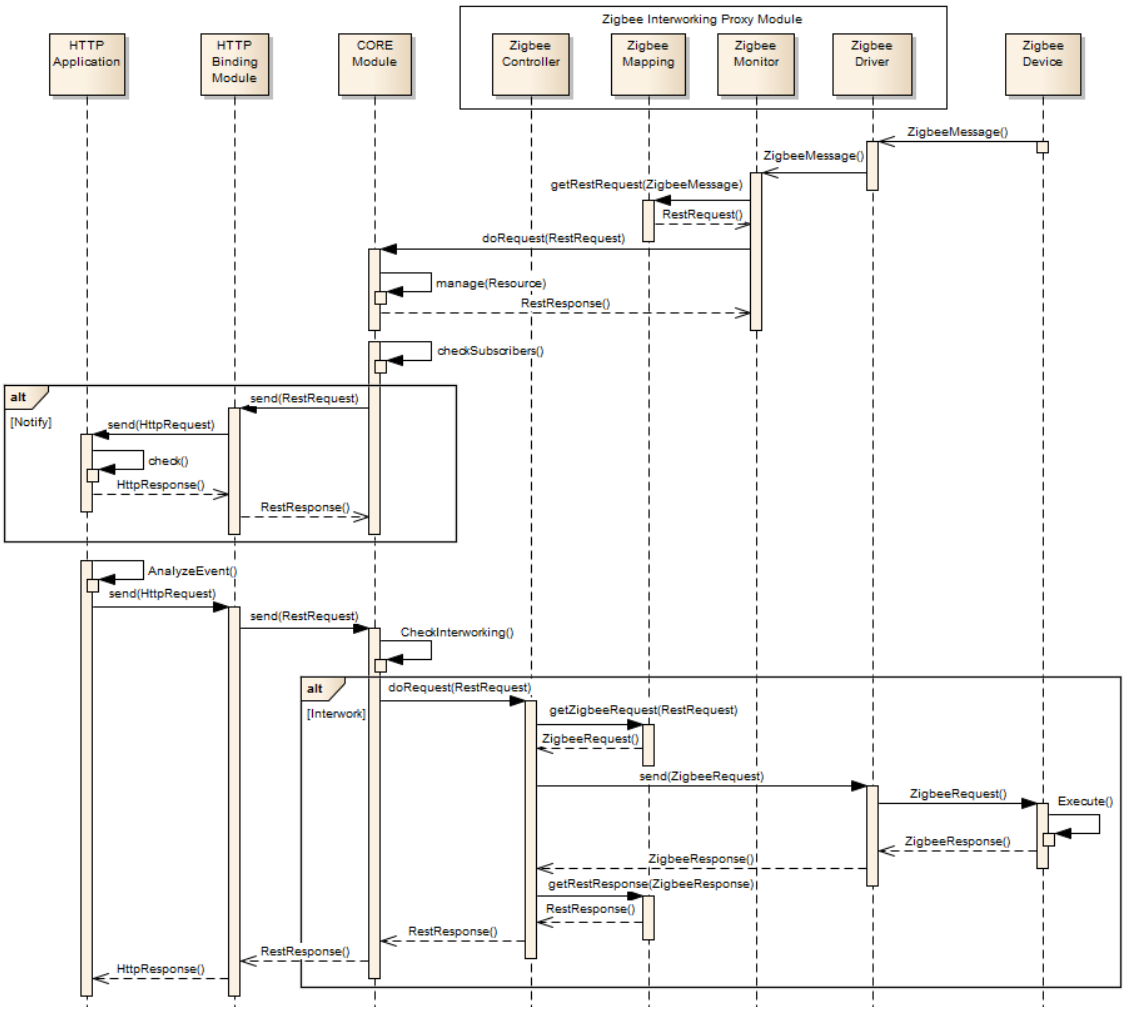

Figure 2.12: Interworking with legacy devices

the Applications, Containers and ContentIntances resources. However, the SmartM2M service platform was designed to meet mainly the common requirement of M2M domains, it does not cover specific aspects related to each vertical domains. The granularity of SmartM2M was limited to the contentInstance resource, which is opaque to the platform. ContentInstances are used as black boxes to store data and exchange information between applications. This is why an additional specific data model is required to accurately describe objects, data, and relations related to the corresponding vertical domain. This is the reason why SmartM2M suggested to use the Open Building Information Exchange (oBIX) to facilitate the exchange of information in the Smart Building domain. The oBIX data model is used to represent the device meta data including device description, available operations, and data they handle.

The root abstraction in oBIX is object, modeled in XML via the “obj” element. A variety

of oBIX types are defined as a derivative of the “obj” element such as “str”, “bool”, “int”, list”, “op”, “date”, “time”, etc. Any “obj” element or its derivatives can contain other “obj” elements. The “obj “element supports attributes such as “name”, “val”, “is”, “href”, etc. The oBIX data model offers a high level of abstraction and flexibility that enables one to accurately represent the device meta data and also the generated data. Figure 2.13 shows an XML example of the content representation of “DESCRIPTOR” contentInstance encoding using oBIX. A single “obj” element is used as a root node to represent the device meta data including both attributes and operations. The “str” element is used to represent every string attribute such as the device manufacturer, serial number, location, etc. The “op” element is used to represent every operation offered by the device. The “name” attribute is used to set the name of the operation such as “STATE”, “ON”, OFF”,and “TOGGLE”. The “href” attribute is used to set the URI of the corresponding operation. We think that the “name”, and “href” attributes of the “op” element are not sufficient to tell the issuer on how to formulate its request. This is why, we propose to use the “is” attribute of the “op” element to inform interested clients about the type of operation. Five operation types inline with the REST style are defined including “retrieve”, “create”, “update”, “delete”, and “execute”.

```
<obj>
    <str name="type" val="Lamp"/>
    <str name="location" val="Home"/>
    <str name="manufacturer" val="XYZ"/>
    <str name="serial" val="123"/>
    ...

    <op name="GetState" is="retrieve" href="GSCL_1/applications/LAMP_1/containers/DATA
        /contentInstances/latest/content" />
    <op name="GetState(Directly)" is="retrieve" href="GSCL_1/applications/LAMP_1/retarget" />
    <op name="SwitchON" is="execute" href="GSCL_1/applications/LAMP_1/retarget/true" />
    <op name="SwitchOFF" is="execute" href="GSCL_1/applications/LAMP_1/retarget/false" />
    <op name="Toggle" is="execute" href="GSCL_1/applications/LAMP_1/retarget" />
    ...
</obj>
```

Figure 2.13: oBIX XML description example

The oBIX data model can be used to represent the data generated by the device such as events or state changes. It can be used to describe the data received by the device as well such as request or actions. Figure 2.14 shows an XML example of the content representation of “DATA” contentInstance encoding using oBIX. A single “obj” element is used as a root node to represent the data payload. A “bool” element is used to represent the value of the boolean device state. The “abstime” is used to represent the timestamp of the generated event.

```
<obj>
    <bool name="data" val="true"/>
    <str name="timestamp" val="2014-04-07T14:45:33"/>
    ...
</obj>
```

Figure 2.14: oBIX XML data example

It is clear that oBIX is a generic solution to structure the payload of M2M applications. However it is not sufficient to solve the semantic interoperability issues to enable machine commutable logic, inference, knowledge discovery, and data federation between M2M applications. In fact, oBIX does not define a dictionary that enable one to represent M2M concepts, attributes and their relations without ambiguity. In oBIX, each application has to personalize the payload using its own vocabulary. So all interacting applications must agree beforehand on a specific terminology before establishing any communications. It means that there is, in the current solution, a strong coupling between applications, which is in contradiction with the main goal of the SmartM2M standard. To overcome the semantic gap, we propose in section 2.9 a semantic model for IoT based on ontology as an alternative to the current oBIX data model.

2.7.9 Visualization interfaces

To make it easy for developers to follow the evolution of resources inside every distributed machine, we extended OM2M features with a web resource browser plugin. This plugin is designed mainly for development and testing and is not dedicated for end users. It uses a web HTTP client to interact with the OM2M REST API in order to discover resources in distrusted machines in a seamless way. It offers the possibility for users to browse resources and manage devices simply by following hypertext links in the same way we browse a simple web site. This interface is capable also to display and configure the right graphical buttons to monitor sensors and control actuators simply by paring the oBIX device representation. Figure 2.15 shows the resource browser plugin web interface. The left side display the resource tree related to a fan actuator. The right side display the fan attributes and discovered services.

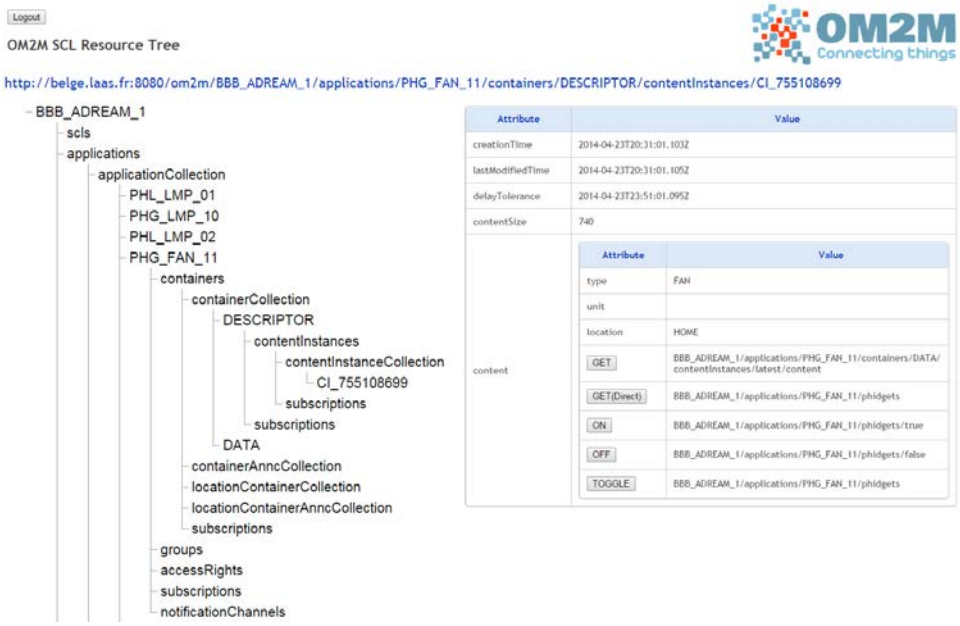

Figure 2.15: Resource tree browser web interface

In addition, to make it easy for end users to discover the list of connected devices, monitor sensors, and control actuators, we extended OM2M features with a web visualization and control plugin. This plugin uses also a web HTTP client to interact with the OM2M REST API and discover existing devices. According to the device description, it selects the appropriate charts diagram, and display the device generated data in a nice graphical way. Figure 2.16 shows the OM2M visualization plugin web interface.

Figure 2.16: Device monitoring web interface

2.8 Enabling constrained devices in SmartM2M

The vast majority of devices that will constitute the IoT are expected to be severely constrained in terms of memory, CPU, and power capacities. Unfortunately, the integration of these constrained devices in the SmartM2M service platform is not straightforward even when using the CoAP communication protocol binding. In fact, there is a deep problem impacting the current SmartM2M resource architecture, which is based on an inefficient naming convention that compromises the mass deployment of M2M services especially in constrained environments. To effectively solve this problem, we propose a non hierarchical resource structure for the SmartM2M architecture, which is better suited for constrained devices [14].

2.8.1 SmartM2M hierarchical resource structure

The SmartM2M standard relies on a descriptive naming convention for the resources in such a way that the meaning of each resource can be clearly conveyed and understood. These predefined

names are considered in the definition of the resource representation, which results naturally in large payload. In order to cope with limitations imposed from constrained devices, an informative annex has been introduced on the second release of the SmartM2M standard to enable short names and small payload. On one hand, a new naming convention is defined that fixes the size of each resource name to three characters. On the other hand, more efficient encoding formats are supported to reduce the overhead of the information payload exchange. However, this optimization is not sufficient to meet the requirements imposed by constrained devices. It can be seen as a workaround that implies that a genuine solution to the problem is needed.

The main problem of the current approach is precisely the hierarchical resource structure, defined in SmartM2M, which generates long URI even when using a short naming convention. Figure 2.17 (a) shows an example about how the resource URIs are defined in the current SmartM2M architecture. The client creates an application resource with the id "AP-ID" by sending a creation request to "/SB/SB-ID/AP", and receives back the location of the created application, which is "/SB/SB-ID/AP/AP-ID". Then, the client creates a container resource with the id "CO-ID" by sending a creation request to "/SB/SB-ID/AP/AP-ID/CO", and receives back the location of the created container, which is "/SB/SB-ID/AP/AP-ID/CO/CO-ID". Finally, the client creates a content instance resource with the id "CI-ID" by sending a creation request to "/SB/SB-ID/AP/AP-ID/CO/CO-ID/CI", and receives back the location of the created content instance, which is "/SB/SB-ID/AP/AP-ID/CO/CO-ID/CI/CI-ID". Now, any issuer who is interested in the created content instance can simply send a retrieval request to "/SB/SB-ID/AP/AP-ID/CO/CO-ID/CI/CI-ID".

In SmartM2M, it is clear that the overall URI length increases in function of the position of the resource in the resource tree. For example, in the case of a contentInstance resource, the relative part of the URI can reach up to eight levels. This is far too large for use with 6LoWPAN nodes, which has a maximum frame size of 128 bytes including only 53 bytes for the application payload. It means that, when using CoAP over 6LowPAN, a long URI will use a large part of the available application payload, making impossible to send the required CoAP options, body, and other parameters in one message. When a node must transmit a packet that is larger than the maximum transmission unit (MTU) of the network, it breaks the packet into several smaller fragments and sends them in separate message instead. Fragmentation should be avoided in 6LowPAN application protocols because it causes serious performance issues due to the low-bandwidth, the lossy network, and the delay of the wireless channel. In fact, lost fragments cause the whole packet to be retransmitted.

2.8.2 Proposed non hierarchical resource structure

To effectively solve this problem, we propose a non hierarchical resource structure for the SmartM2M architecture that relies on flat URIs, which is more suited to be used in constrained devices. Our new approach offers a flexible naming convention aiming to limit the number of nested resources and therefore optimize the overall URI length. The main idea behind the proposed solution consists in using the same URI template for all resources in such a way they become addressable in a short and homogeneous way. Such template should be sufficiently generic that it can cover all the primitive procedures defined in the SmartM2M Standard. The following template is defined to meet this goal:

- /RESOURCE_TYPE/RESOURCE_NAME{/COLLECTION_TYPE}

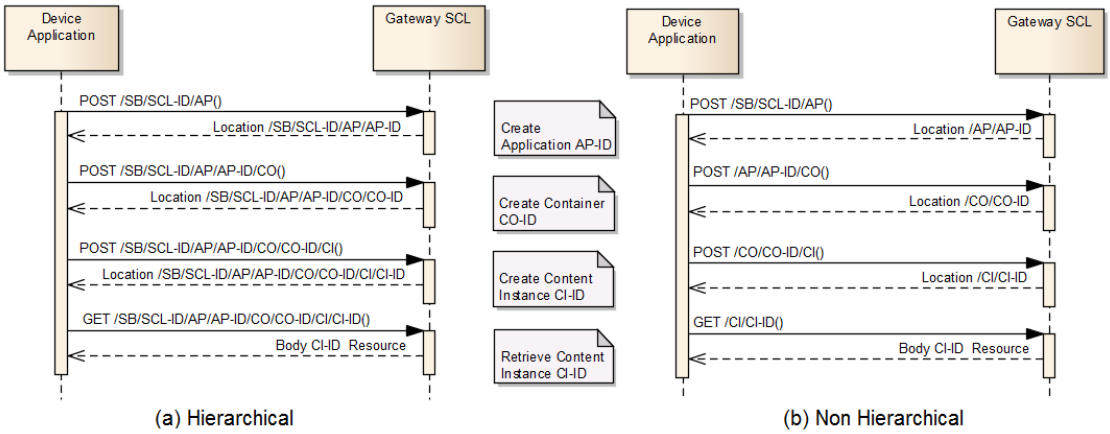

Figure 2.17: Comparison between Hierarchical and non Hierarchical URI

According to this template, each resource URI is composed of three parameters. The first parameter "RESOURCE_TYPE" is mandatory. It informs about the type of the addressed resource, which makes the URI understandable for the client. Additionally, it is important to keep the resource type in the URI because it enables one to reduce the server processing overhead, and so improve the system performance. In fact, the resource type parameter enables the server to route requests of retrieval, update, or delete to the appropriate resource controller simply by looking into the request URI and the method. The second parameter "RESOURCE_NAME" is mandatory and indicates the addressed resource name. In our approach, the uniqueness of the resource name is required only between resources of the same type. The third parameter "COLLECTION_TYPE" is optional and is used to access available collections pertaining to the targeted resource. In our approach, a collection is a virtual resource that is not persisted in the database. The collection representation should be computed dynamically. This parameter is specified because it makes the URI clearer for the client. It is useful for the server to enhance its performance as well. In fact, the collection type parameter enables the server to route requests of creation to the appropriate resource controller without the need to parse the received resource representation.

We can distinguish two possible applications of the proposed template. On one hand, the following specific template "/RESOURCE_TYPE/RESOURCE_NAME" can be used to retrieve, update, or delete a resource named "RESOURCE_NAME". On the other hand, the following specific template "/RESOURCE_TYPE/RESOURCE_NAME/COLLECTION_TYPE" can be used to retrieve, update, or delete a collection pertaining to a specific resource named "RESOURCE_NAME", or to create a new resource within the collection "COLLECTION_TYPE".

To better understand how this new approach works, let's take a concrete example. In analogy to the SmartM2M example already described in Figure 2.17 (a), Figure 2.17 (b) shows the same scenario using the new approach. This figure describes how the new flat resource naming convention is used according to the proposed template. The client creates an application resource with the id "AP-ID" by sending a creation request to "/SB/SB-ID/AP", and receives back the location of the created application, which is "/AP/AP-ID". Then, the client creates a container resource with the id "CO-ID" by sending a creation request to "/AP/AP-ID/CO", and receives back the location

of the created container, which is "/CO/CO-ID". Finally, the client creates a content instance resource with the id "CI-ID" by sending a creation request to "/CO/CO-ID/CI", and receives back the location of the created content instance, which is "/CI/CI-ID". Now, any issuer who is interested in the created content instance can simply send a retrieval request to "/CI/CI-ID", which is very short compared to "/SB/SB-ID/AP/AP-ID/CO/CO-ID/CI/CI-ID" required for SmartM2M.

2.8.3 Resource naming conventions evaluation

In this section we evaluate the performance of the proposed solution compared to current approaches already defined in SmartM2M. Our non-hierarchical short name naming convention is compared to both SmartM2M hierarchical long name and SmartM2M hierarchical short name structures. Two concrete scenarios bringing into play an interaction between a device and a gateway are considered for this evaluation. The first scenario is focusing on the device registration phase to measure the size of the URI for creation requests. The second scenario is focusing on resource discovery to measure the size of the URI for retrieval requests. Resources like Application, Container, and ContentInstance are considered in both scenarios. For sake of simplicity, we set a unique size of 8 characters for the generated resource ID.

The results of the first scenario dealing with resource creations are described in Figure 2.18. It is clear here that the hierarchical long name approach generates the longest URI and can reach 76 characters for the content instance creation. So we will focus more on the two other approaches. For the first request, which is related to application creation, the short name Hierarchical and non hierarchical solutions generates the same URI size equal to 15 characters because the request is sent to the root of the resource tree. However, when we go deeply into the resource tree for create, the URI size increases in the hierarchical approach short name. It requires 27 characters to create the Container, and 39 characters to create the contentInstace. For the non hierarchical approach, the size of the URI remains constants and does not exceed 15 characters.

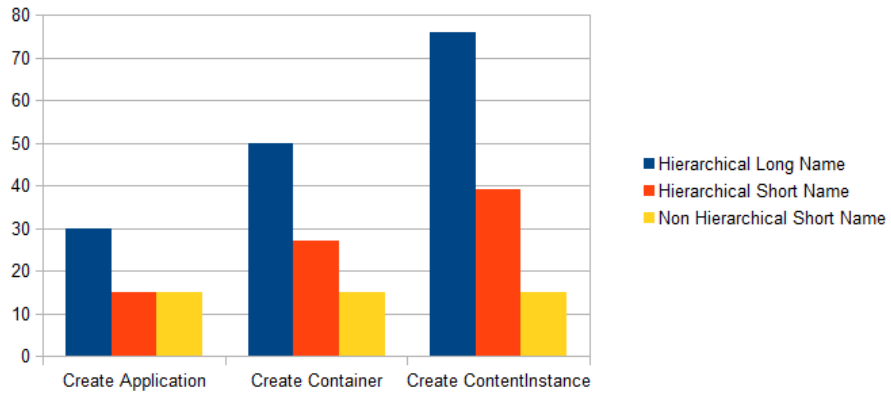

Figure 2.18: Create Evaluation

The results of the second scenario dealing with resource retrievals are described in Figure 2.19. The result are very similar to the measurement related to creation requests. The long name ap-

proach generates the longest URI and can reach 76 characters for the content instance creation. The hierarchical long name approach generates the longest URIs and can reach 85 characters for contentInstance retrieval. The results are not much better for the hierarchical short name and 25 characters are required to retrieve application, 32 characters to retrieve container, and 47 characters to retrieve contentInstance. However, the non hierarchical solution generates a constant URI size that does not exceed 12 characters.

Results obtained for the retrieval requests are applicable to delete and updates. So this evaluation covers main operations and resources. In the proposed solution, the size of the URI remains constant for any type of resource and any type of operation. This can be explained by the fact that the proposed naming convention decouples the URI structure from the hierarchy of the resource, and so enables one to interact with a deeply hierarchical resource structrue using very short and flat URI.

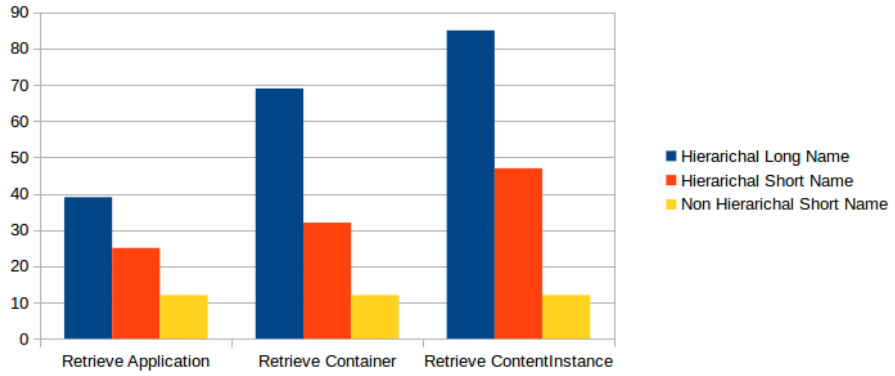

Figure 2.19: Retrieve evaluation

2.9 IOT-O ontology for semantic data interoperability

SmartM2M did not standardize the device data model and the data they handle. It was limited to some interworking use cases based on the oBIX format. In this section, we propose the IoT-O ontology [15] to enable semantic interoperability in M2M systems. The IoT-O ontology aims to represent IoT concepts and relationships independently from the underlying formalism originally used for describing them. Through comprehensive uses cases, we show the use of IoT-O along the SmartM2M standard.

2.9.1 Towards semantic IoT

Ontologies have proven to be beneficial for intelligent information integration, information retrieval, and knowledge management. They enable one to index resources content using semantic annotations that can result in the representation of explicit knowledge that cannot be accessed and managed. Ontologies are very popular and useful for overcoming challenges fixed in the proposed study because

they provide an efficient way of cleverly structuring a domain making use of semantic hierarchical and property/value relationships based on a vocabulary of concepts/instances [4].

In an M2M system, users and applications should be able to discover, monitor and control sensors and actuators offering particular services and having particular properties with a high degree of automation. To reach this goal, an ontology for IoT shall represent a variety of concepts such as platform, deployment system, thing, device, manager, service, sensor, actuator, sensing and actuating capabilities, observation, operation, time, unit, kind, value, and other related concepts with their relationships.

Since ontologies are designed to be reusable and extensible, it is possible to define a complete ontology for IoT by reusing existing ontologies. New concepts should be designed only when needed. This approach enables one to reduce the ambiguity of IoT terminology and allows rapid convergence to a common vocabulary.

2.9.2 IoT-O ontology model

Since there is no single model that covers all IoT concepts and their relationships, a set of well defined ontologies were carefully selected in order to form an efficient ontology for IoT. Figure 2.20 shows how the selected ontologies are merged together to form the M2M ontology. The M2M ontology consists of five main parts, which are sensor, observation, actuator, actuation and service models. The DUL upper ontology was selected to describe very general concepts that are the same across all knowledge domains, and so facilitate reuse and interoperability. It is a lightweight foundational model for representing either physical or social contexts. The SSN ontology, which is aligned with DUL, was selected to represent sensors in terms of measurement capabilities and properties, observations and other related concepts. However it does not describe actuator devices.

2.9.3 Use cases

Since currently there is no ontology that accurately describes actuators, we designed a new ontology called ACT, which is inspired from SSN and aligned with DUL, to describe actuators in terms of actuating capabilities and properties, actuation, and related concepts. The QUDV ontology was selected to represent quantities, units, dimensions and values. The OWL-TIME ontology was selected to provide a vocabulary for expressing facts about topological relations among instants and intervals, together with information about duration, and about date time information.

Given that SmartM2M aims to enable seamless interactions between business applications and services, it is important to represent how these services can be requested, without any ambiguity in order to reduce the amount of manual effort required for discovering and using them. The MSM ontology was selected to describe services since it provides a common vocabulary based on existing web standards able to capture the core semantics of both Web services and Web APIs in a common model. Each service is described using a number of operations that have address, method, input and output MessageContent descriptions. To understand how the M2M ontology works, let's consider a concrete example representing a real sensor and actuator using the M2M ontology.

The PHIDGETS_1124 temperature sensor is an inexpensive sensor from Phidgets. It is connected to an analog input and measures the ambient temperature in the area surrounding the board with a range of -30 Celsius to +80 Celsius with a typical error of 0.5 Celsius. Each measurement includes the temperature value in Celsius, a timestamp, and a measurement quality ranging from 0 to 5. The sensor offers a web service to enable remote temperature access. The temperature

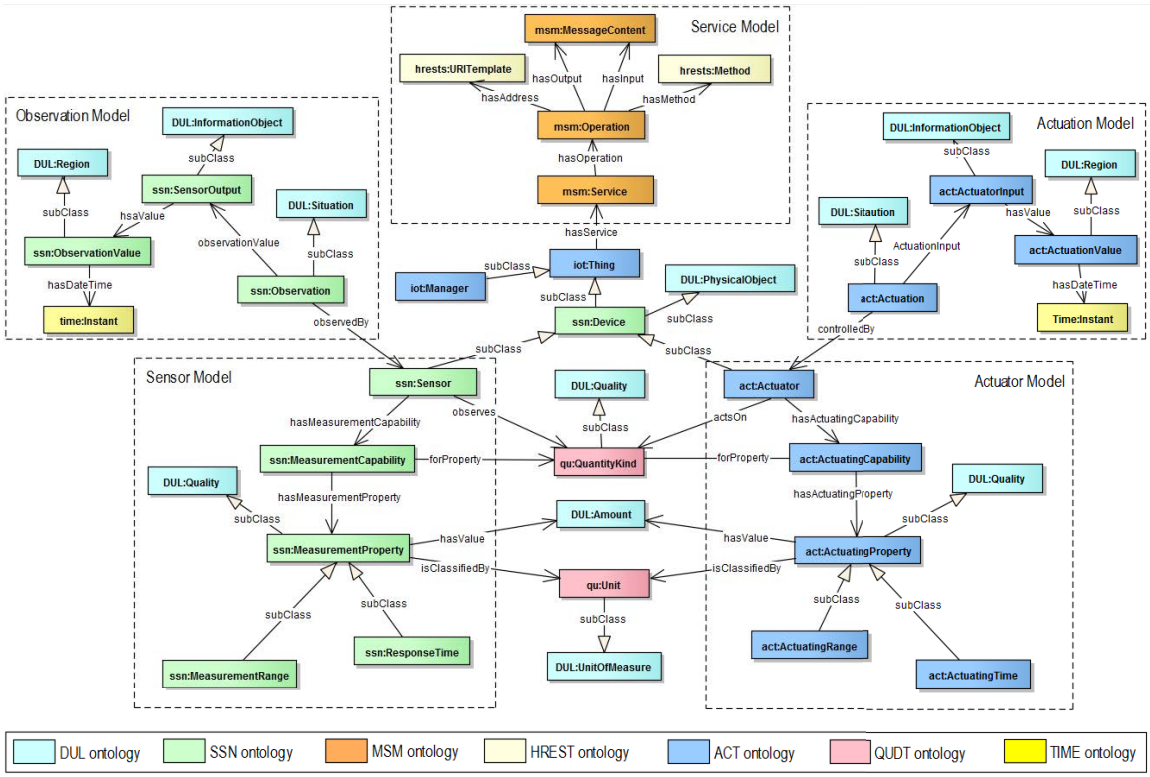

Figure 2.20: Ontology model for M2M

can be retrieved instantaneously by sending a retrieve request to the address "gscl/applications/PHIDGETS_1124/retargeting".

Figure 2.21 details the corresponding ontology instance. It shows how the sensor, observation, and service information are inserted in the M2M ontology. The sensor model represents the temperature sensor information and measurement capabilities including measurement range and accuracy. The Observation model represents the temperature data including the observed value, the timestamp and the quality of observation. The Service model represents the temperature web service including the state retrieve operation, address and method.

The HUELUX actuator is a digitally dimmable wireless lighting bulb from Philips. It has a power range of 0 Watt to 50 Watt with a lighting time of 2 seconds. The illuminance level can be dimmed by requesting the required power value. The light bulb offers a web service to enable remote illuminance control. The illuminance can be dimmed instantaneously by sending a create request to the address "gscl/appliactions/HUELUX_APP/retargetting" with a message body containing the required power.

Figure 2.22 details the corresponding ontology instance. It shows how the actuator, actuation and service information are inserted in the M2M ontology. The actuator model represents the light bulb information and actuating capabilities including power range and lighting time. The actuation model represents the dimming command. The Service model represents the light bulb web service

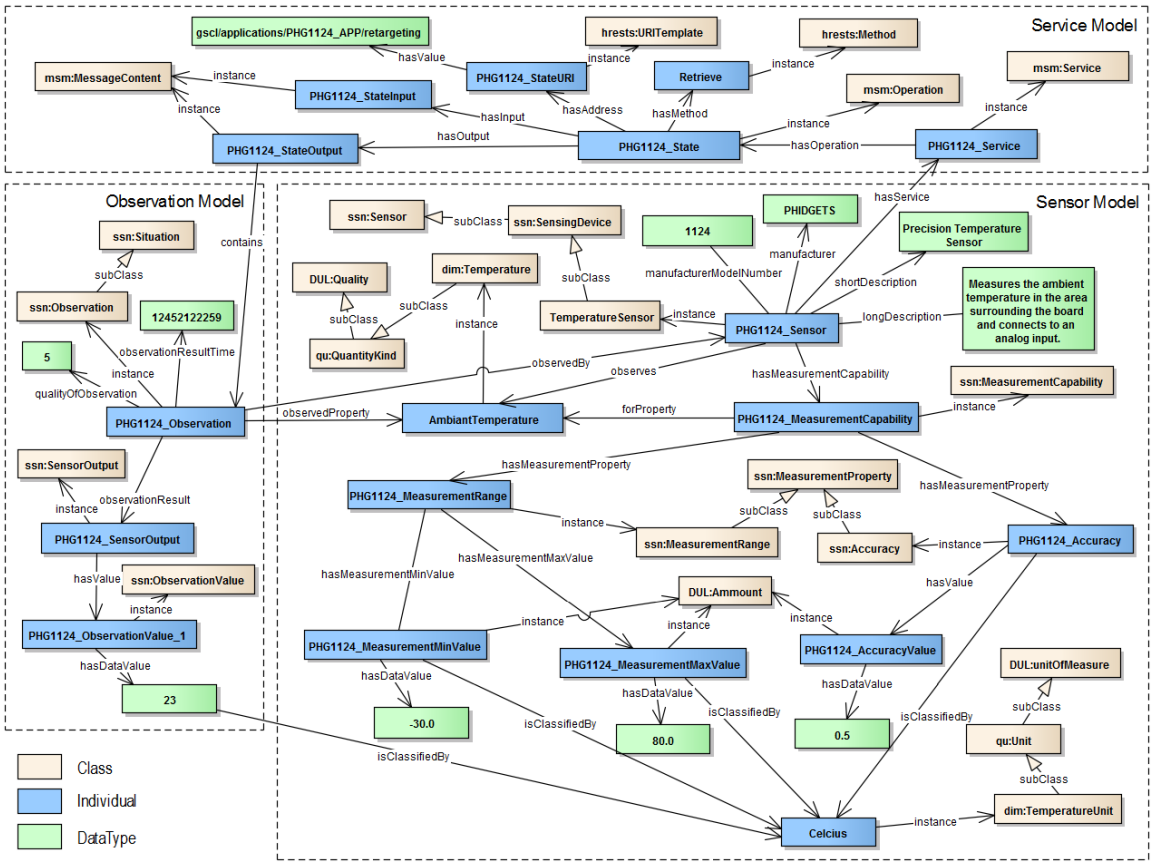

Figure 2.21: Temperature sensor model example using the M2M ontology

including the illuminance dimming operation, address and method.

2.10 Conclusion

In this chapter, we presented the fundamental concepts of IoT and M2M, and listed the main application domains and most important challenges facing M2M. In addition, most popular architecture styles are compared and related standardization and research works addressing the interoperability issues are discussed. A specific section was dedicated to the SmartM2M standard due to its importance and maturity. Based on SmartM2M, the OM2M platform is presented as a solution for the interoperability challenge. We described the OM2M architecture design by detailing its core module features, service discovery, and main extensions. Then, we presented a new resource convention naming with better performance especially in constrained environments. Finally, we described the IoT-O ontology concepts and relationships as a solution for semantic data interopeability and illustrated it through examples.

The OM2M platform made a step forward towards M2M horizontality making possible to in-

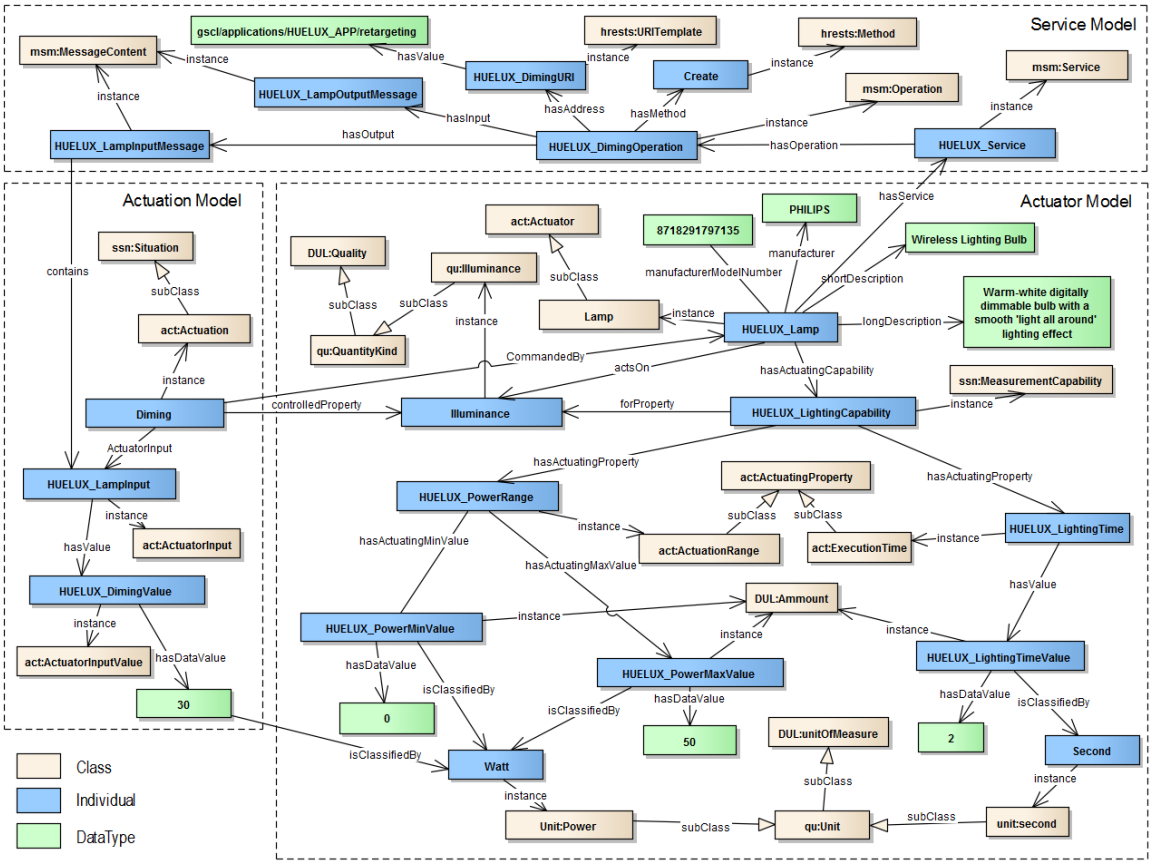

Figure 2.22: Lamp actuator model example using the M2M ontology

terconnect a myriad of heterogeneous device and applications in one system. At the same time, keeping such highly evolving environment alive is costly in terms of time and money. In the next chapter, we focus on extending the OM2M platform with self-management capabilities and semantic reasoning to reduce system complexity based on the Autonomic Computing paradigm.

Chapter 3

FRAMESELF: Self-managing M2M systems

Contents

3.1 Introduction 61
3.2 Autonomic computing paradigm 62
3.3 Self-management approaches 63
3.4 Self-management techniques 65
3.5 FRAMESELF: Autonomic M2M framework 65
3.6 Home automation self-configuration use case 72
3.7 Semantic matching and resource reconfiguration 77
3.8 Conclusion 85

3.1 Introduction

This chapter presents an overview of the Autonomic Computing paradigm and discusses existing approaches and related techniques addressing the increasing complexity challenge of distributed system. We propose the FRAMESELF framework as a solution to retrofit self-management capabilities into M2M system. FRAMESELF comes with a concise and modular autonomic manager architecture and refined data model that extends existing technologies and standards. It is integrated into OM2M as a new service to adapt the system behavior according to the environment changes. A first scenario is illustrated to explain how simple knowledge and syntactic rules can be used with OM2M for self-configuration of the home energy consumption. A second scenario is described to show how the IoT-O ontology can be used with semantic rules to enable OM2M adapt the resource architecture for applications dynamic discovery of applications.

3.2 Autonomic computing paradigm

The autonomic computing paradigm was introduced by IBM in 2001 [17]. Autonomic computing is a concept inspired from biological systems that aims to develop systems capable of self-management to overcome the rapidly growing complexity of computing systems management and to reduce the barrier that complexity poses to further growth. Autonomic managers are the basic building blocks of autonomic systems, which through their mutual interactions produce the overall self-managing behavior of autonomic computing systems. There are four major self-management properties that enable autonomic capabilities in a system [17].

- Self-Configuration is the ability of the system to perform configurations according to predefined high level policies and seamlessly adapt to change caused by automatic configurations.
- Self-Optimization is the ability of the system to continuously monitor and control resources to improve performance and efficiency.
- Self-Healing is the ability of the system to automatically detect, diagnose and repair faults.
- Self-Protection is the ability of the system to pro-actively identify and protect itself from malicious attacks or cascading failures that are not corrected by self-healing measures.

Since the introduction of the term autonomic computing, several efforts have been done in academia and industry to build systems with autonomic capability [58, 59]. These works define infrastructures that mostly focus on one of the two design approaches, Externalization and Internalization [60]:

- Externalization Approach: modules enabling self-management lie outside the managed system.
- Internalization Approach: application specific self-management is done inside the managed system.

The evolutionary path to autonomic computing is represented by five levels [61], starting from basic, through managed, predictive, adaptive and finally to autonomic:

- Basic Level: At this level, each system element is managed by IT professionals. Configuring, optimizing, healing, and protecting IT components are performed manually.
- Managed Level: At this level, system management technologies can be used to collect information from different systems. It helps administrators to collect and analyze information. Most analysis is done by IT professionals. This is the starting point of automation of IT tasks.
- Predictive Level: At this level, individual components monitor themselves, analyze changes, and offer advices. Therefore, dependency on persons is reduced and decision making is improved.
- Adaptive Level: At this level, IT components can individually or group wise monitor, analyze operations, and offer advices with minimal human intervention.

- Autonomic Level: At this level, system operations are managed by business policies established by the administrator. In fact, business policy drives overall IT management, while at adaptive level; there is an interaction between human and system.

As described in Figure 3.1, an autonomic manager is organized into four main modules, which are Monitor, Analyzer, Planner and Executor. These modules share a same knowledge (managed resources details, policies, symptoms, request for change, plans, etc.), exploit policies based on goal and environment awareness and constitute together the MAPE-K control loop to dynamically manage entities using sensors and effectors.

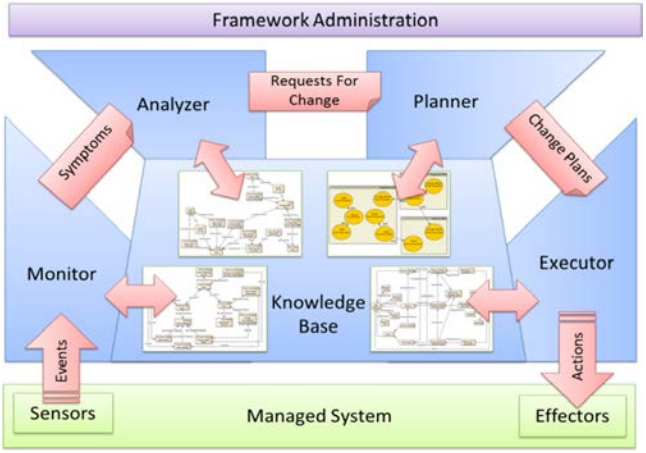

Figure 3.1: MAPE-K loop reference architecture

An autonomic manager may detect a situation with which it is not familiar, it then has the option to forward this situation via its sensor interface to another autonomic manager. Also, it may not have control or access to specific resources it requires to complete a plan. It then need to forward a request for more resources, identifying itself in such a way that it can be notified via effectors interface when these resources become available. The autonomic manager should be able to advertise its services and act as delegate for another autonomic manager [62].

3.3 Self-management approaches

Autonomic computing has been inspired from biological systems. Both centralized and decentralized approaches have been suggested to enable autonomic behavior in computer systems [12, 63]. The centralized approach focuses on the role of human nervous system as a controller to regulate and maintain the other systems in body [64]. Decentralized approaches take inspiration from local and global behavior of biological cell and ant colony networks [65].

3.3.1 Distributed databases

Distributed databases focuses on developing self-managed large scale systems. One such project is Oceano [66], a project of IBM that provides a pilot prototype of a scalable, manageable infrastructure for a large scale utility powerplant. OceanStore [67], a project of University of California

Berkeley describes a global persistent data store for scaling billion of users. IBM's SMART [68] offers a solution to reduce complexity and improve quality of service through the advancement of self-managing capabilities. Similarly, Microsoft's AutoAdmin [69] makes database system self-tuning and self-administering by enabling them to track the usage of their system and adapt to application requirements.

3.3.2 Agent architectures

Agent architectures have been frequently used to develop infrastructures supporting autonomic behavior based on a decentralized approach. Unity [70] makes use of this approach to achieve goal-driven self-assembly, self-healing and real-time self-optimization. Similarly, Autonomia [71], a project of University of Arizona provides required tools to specify the appropriate control and management schemes for maintaining quality of service requirements.

3.3.3 Component based frameworks

Component based frameworks have been suggested for enabling autonomic behavior in systems. The Accord framework [72] facilitates development and management of autonomic applications in grid computing environments. It provides self-configuration by separating component behavior from component interaction. Selfware [73], models the managed environment as a component based software architecture, which provides means to configure and reconfigure the environment and implements a control loops, which link probes to reconfiguration services and implement autonomic behaviors.

3.3.4 Artificial intelligence and control theory

A number of frameworks make use of techniques such as artificial intelligence (AI) and control theory. In [74] an AI planning framework based on the predicate model has been presented. It achieves user defined abstract goals by taking into account current context and security policies. Another area focuses on modifying control theory technique [75] and presents a framework to address resource management problem. A mathematical model is used for forecasting over a limited time horizon.

3.3.5 Service Oriented Architecture

Service Oriented Architecture has been also used to design architecture with autonomic behavior. In [76], autonomic web services implement the monitoring, analysis, planning and execution life cycle. The autonomic web service uses log file provided by a defective functional service to diagnose the problem and consults the policy database to apply appropriate recovery action.

3.3.6 Autonomic infrastructures

Infrastructures have been proposed to inject autonomic behavior in legacy and non-autonomic system, where design and code information is unavailable. Some of the frameworks identified, make use of layered architecture [77], case based reasoning [78], and a rule driven approach [79] to enable autonomicity in existing systems.

3.4 Self-management techniques

Since autonomic computing enables one to decrease the system maintenance costs in terms of time and money, many industrials and researchers developed new technologies to improve their current products and provide new experiments and techniques.

3.4.1 Self-configuration

There are more system specific techniques for self-configuration such as hot swapping [80] and data clustering [81], and techniques that make use of more generic approaches such as machine learning such as ABLE [82] and case-based reasoning [78]. Other studies rely on code injection mechanisms [83] to inject self-configuration properties into object oriented system using a proxy/wrapper architecture.

3.4.2 Self-optimization

Many applications that require continuous performance improvement and resource management make use of control theory approach demonstrated in the Lotus Notes application [84], learning based self-optimization techniques such as LEO [85], and active learning based approach [86] based on building statistical predictive models.

3.4.3 Self-healing

Case base Reasoning (CBR) [78] was used for self-healing to perform problem detection in the analysis phase, find a solution in planning phase and performs problem repair in execution phase. Hybrid approach [87] enables one to create a survivability model provides more robust services. Active probing and Bayesian network [88] allow probes to be selected and sent on demand. The heartbeat failure detection algorithm [89] detects problem when the monitor experiences a delay in receiving of message. Tools like Pip [90] exposes structural errors and performance problems by comparing actual system behavior and expected system behavior. Rx [91] rolls back the program to a recent checkpoint upon a software failure, and then re-executes the program in a modified environment.

3.4.4 Self-protecting

To enable self-protecting, some techniques make use of component models such as Jade [92] and self-certifying alerts as demonstrated in Vigilante [93]. [94] proposes to use differing implementations of software and hardware in order to reduce the potential total damage from a single exploit.

3.5 FRAMESELF: Autonomic M2M framework

Creating highly distributed systems that manage themselves using specific approaches and techniques in accordance with high-level policies is possible. However, inventing new technologies is not enough to meet the autonomic computing challenge. What is missing here is a refined architectural approach for self-management that works in a multi-vendor environment and consolidates current efforts for a better performance result.

In this section, we propose FRAMESELF [13, 15, 19–21] to create autonomic M2M systems. FRAMESELF provides a concise architecture and management data that extends the current MAPE-K loop standard and exploits the existing technologies and standards.

3.5.1 Component-based architecture

The FRAMESELF architecture is based on the MAPE-K loop reference model, and so composed of monitor, analyzer, planner, and executor modules that share a same knowledge base and operate together to manage a specific target system. These modules act as experts to emulate the decision-making ability of humans and are designed to solve complex problems by reasoning about knowledge. Each module is composed of a set of well-defined rule-based components and is divided into two parts: the inference engine and the knowledge model. In addition, a mapping infrastructure is defined to support different managed system and interwork with specific sensors and actuators. A management interface is defined to administer the MAPE-K modules and to follow the execution of the self-management operations as well. A detailed components diagram is illustrated in Figure 3.2 to show the structure of the proposed autonomic architecture solution.

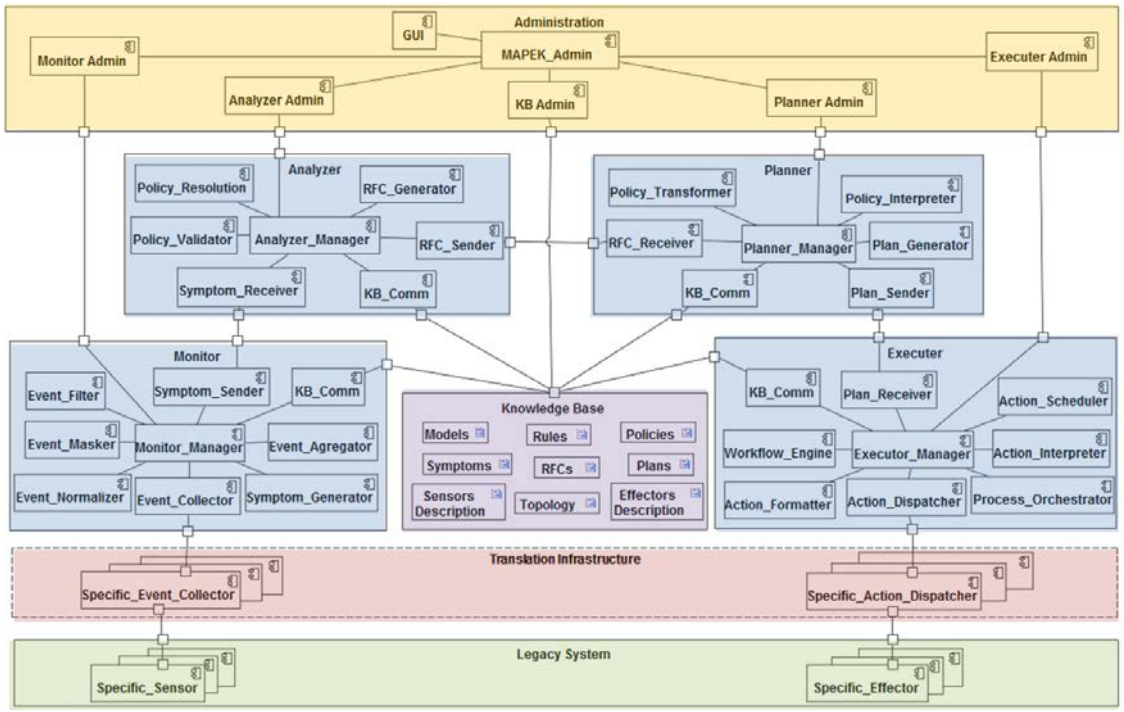

Figure 3.2: FRAMESELF components diagram

Monitor

The monitor module answers to the "what is happening?" question. It collects events from managed resources by means of sensors, updates a model describing the sensors environment and topology

located on the knowledge base with relevant information from received events, and infers new knowledge about symptom occurrences, then extracts relevant information and sends them to the analyzer. The monitor component contains the **Monitor_Manager** that controls sub-components for collecting, normalizing, masking, filtering, and aggregating events. It also manages sub-components for symptom generation and symptom sending. The **Event_Collector** collects events from different sensors. The **Event_Normalizer** transforms, extends and formats captured events in a standardized event format to make them consistent for processing. The **Event_Filter** discards events that are deemed to be irrelevant for the management platform. The **Event_Masker** ignores events pertaining to systems that are downstream of a failed resource. This is different from filtering because masking allows one to dynamically hiding unfiltered events according to the context changes. The **Event_Aggregator** merges duplicates of the same event that may be caused by network instability. The **KB_Comm** enables one to read/write information related to monitoring operations from/to knowledge base. The **Symptom_Generator** reasons about monitoring knowledge models to infer symptoms based on a list of events. The **Symptom_Sender** sends generated symptoms to the Analyzer.

Analyzer

The analyzer module focuses on the "what to do?" question. It provides mechanisms that correlate and model complex situation. These mechanisms allow the autonomic manager to learn about the environment and help to predict environment changes. The analyzer receives symptoms as input, uses them to update a knowledge model describing the complex situation. The inference engine generates new knowledge about required Requests For Change (RFCs), and sends them to the planner. The Analyzer component includes the **Analyzer_Manager** that uses sub-components for symptom reception, and RFCs generation and sending. It manages also sub-components for policies resolution and validation. The **Symptom_Receiver** subscribes to the message bus and receives published symptoms. The **KB_Comm** enables one to read/write information related to analyzing operations from/to knowledge base. The **RFC_Generator** reasons about knowledge models to infer RFCs based on a symptom list. **RFC_Sender** publishes generated RFCs to the message bus to make them available to the planner. The **Policy_Validator** checks and validates a policy entered by the administrator. The **Policy_Resolution** detects and resolves conflicts between a list of policies.

Planner

The Planner module acts as a decision module and focuses on the question "how to do?". It saves received RFCs as goal states, reads models of possible actions and facts from the knowledge base and checks policies to guide its work. Then, it selects actions leading to the goal states and sends them to the executer. The Planner components is based on the **Planner_Manager** that manages sub-components for RFCs reception and Plan generation. It also contains sub-components for policy transformation and interpretation. The **RFC_Receiver** subscribes to the message bus and receives published RFCs. The **KB_Comm** enables one to read/write information related to policies and available effectors from/to knowledge base. The **Plan_Generator** generates action plans leading to received RFCs based on existing effectors and policies. The **Plan_Sender** enables one to publish generated plans to the message bus to make them available to the Executer. The **Policy_Interpreter** processes a policy and employ it to guide the decisions that affect the

autonomic manager behavior. The **Policy_Transformer** takes the policies entered by the system administrator and converts them into low level technology format.

Executer

The Executor module receives as input logical description of the sequence of actions to be executed, and consults a model containing actuators description and available operations details. It matches actions with their correspondent concrete operations, then performs the plan using actuators and controls the sequence of actions execution with consideration for dynamic updates. The executor must answer to the question "how is it done?" by generating reports and saving relevant information into the knowledge base. The Executer contains the **Executer_Manager** that controls sub-component for plan reception, and actions interpretation, scheduling and dispatching. It uses also a workflow engine and process orchestration. The **Plan_Receiver** subscribes to the message bus and receives published plans. The **Action_Interpreter** focuses on processing received actions and checking whether the executor has the rights and means to execute them. The **Action_Scheduler** determines, for simultaneous actions, which action to be executed next to load balance the system. The **Workflow_Engine** describes the automated arrangement and coordination of a list of actions in time. The **Process_Orchestrator** orchestrates the execution of a list of workflows in time. The **KB_Comm** enables one to read/write information related to the operation execution from/to knowledge base. The **Action_Dispatcher** performs actions on different destinations and acts as a proxy to connect specific action dispatchers depending on existing actuators of the targeted legacy system.

Knowledge base

The knowledge base contains all the information needed to perform successful self-management operations on a targeted system. It contains the topology of the managed system architecture and descriptions of existing sensors and actuators that can be entered manually by an administrator, dynamically discovered using a discovery mechanism, or using the control loop by taking into consideration specific "hello events". It also contains information of possible symptoms, RFCs, and actions, as well as policies that can be updated dynamically at run-time. Other domain specific models such as ontology or graphs, and reasoning rules can be instantiated, within the knowledge base, independently for each control loop component. Thus, each model can be handled in a decoupled way, which enable one to manage a multi-model knowledge base.

Translation infrastructure

This infrastructure contains specific components that help to mediate the mapping of events from the sensors to the monitor and the mapping of actions from the executor to actuators. On one hand, the Event_Collector acts as a proxy to connect specific event collectors that depend on the existing sensors. Depending on the targeted system, a Specific_Event_Collector can be deployed as an asynchronous event subscriber able to receive event from an event broker, a synchronous event service requester, or an event log file parser. On the other hand, the Action_Dispatcher acts as a proxy to connect specific action dispatchers that depends on existing actuators. Depending on the targeted system, a Specific_Action_Dispatcher can be deployed as a web-service client, a message publisher, a script launcher, or any other specific mechanism.

Administration

An administration component is designed to enable a human to administer the control loop steps via a graphical user interface. The MAPEK_Admin centralizes all the management functionalities and provides them to the administrator through the GUI sub-component. The Monitor_Admin connects to the Monitor_Manager and allow to configure the Monitor sub-components settings. In particular, it enables one to manage the deployment of Specific_Event_Collector components and managing their life-cycle within the translation infrastructure. The Analyzer_Admin links to the Analyzer_Manager and enables one to configure the Analyzer sub-components parameters. The Planner_Admin connects to the Planner_Manager and allow to acts on the Planner sub-components parameters. The Executer_Admin binds to the Executer_Manager connects to the Executer_Manager and allow to configure the Executer sub-components settings. In particular, it enables one to manage the deployment of Specific_Action_Dispatcher components and managing their life-cycle within the translation infrastructure. The KB_Admin offers a direct connection to the knowledge base and enables one to create, update or delete knowledge information for example entering new effectors descriptions or modifying policies.

3.5.2 Management data elements

The control loop is divided in four main steps, which are monitoring, analyzing, planning, and executing. In each step, a particular type of management data elements is considered. First, the monitor collects events to generate symptoms, then the analyzer handles symptoms to generate RFC, then the planner plans RFC to generate action plans, and finally the executor performs action plan before moving to the next loop. In FRAMESELF, we propose an optimized structures for an event, symptom, RFC, and action, which are inspired from the web event format (WEF) and Symptom Automation Framework (SAF) works, Stanford Research Institute Problem Solver (STRIPS), and Action description language (ADL). Figure 3.3 shows the proposed data structure defined for each element involved in the management process.

Common data

The event, symptom, RFC, and Action elements share the following common attribute:

- Id: A unique name identifying an element,
- Name: Identifies, not necessarily uniquely, an element,
- Category: Classification of elements in order to simplify their processing.
- Description: An explanation of the element in a human-readable format.
- CreationTime: Date-time when an element is created.
- ExpirationTime: Date-time beyond which the element may no longer be useful.

Event

An event represents a significant change in the state of a system resource, network resource or network application. An event can be generated for a problem, for the resolution of a problem or for the successful completion of a task.

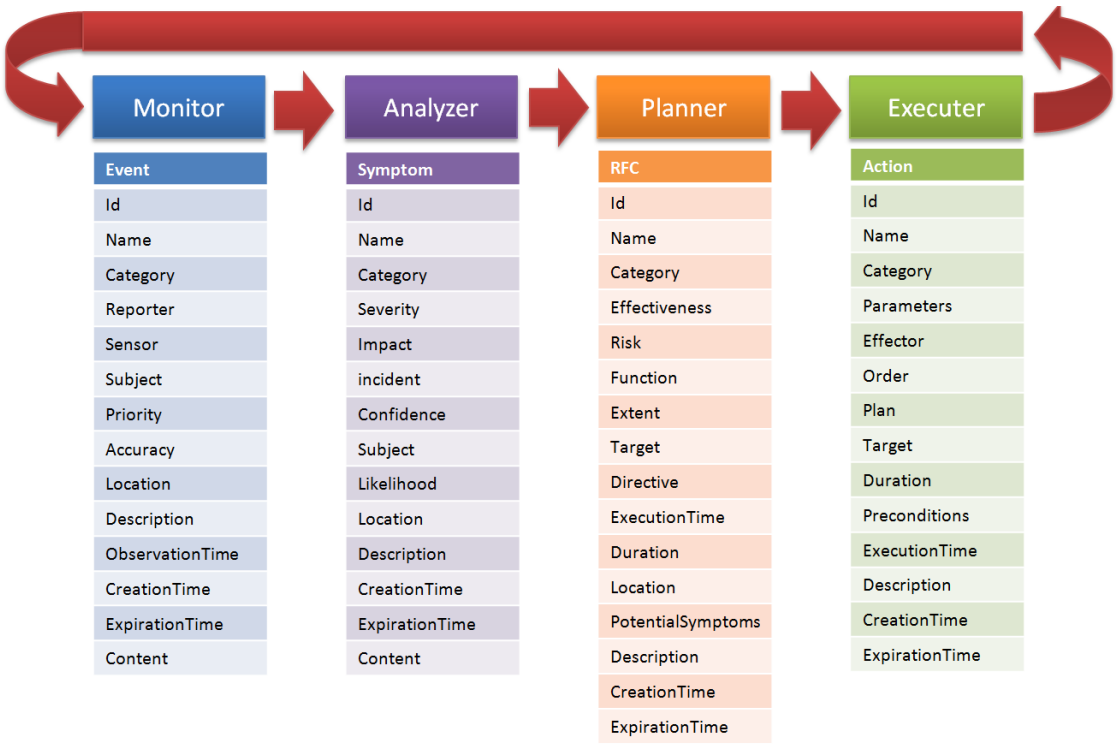

Figure 3.3: FRAMESELF data format

- Priority: Importance of an event.
- Accuracy: Quality accuracy level of an event as determined by the sensor.
- Reporter: Identification of the entity that emitted the event.
- Sensor: Identification of the entity that measured the event.
- Subject: Identification of the entity related to the event.
- ObservationTime: Date-time when the event is measured.
- Location: Identification of the location where the event is measured.
- Content: Raw value of the event.

Symptom

A symptom is a form of knowledge that indicates a possible problem or situation in the managed environment. Symptoms are recognized in the monitor component of the control loop, and used as a basis for an autonomic analysis of a problem or a goal.

- Severity: Perceived severity of the status the event is describing with respect to the application that reports the event.
- Impact: Effect of this symptom with respect to the consequences of not treating it.
- Incident: Groups a set of symptoms that are related to the same incident.
- Likelihood: Typicality of a given symptom.
- Confidence: Level of confidence in the generation of a given symptom.
- Subject: Identification of the entity affected by the symptom.
- Location: Identification of the location affected by the symptom.
- Content: Raw value of the symptom.

Request For Change

A Request for change is a proposal for the implementation of a change. It provides remediations, diagnostics, preventive measures, or optimization to be performed and may be applied to any component or any aspect of an M2M system. A RFC should contain all information required to approve a Change. Further information can be added as the Change progresses through its life-cycle.

- Effectiveness: The expected success of the request.
- Risk: The expected side effects or undesired consequences of performing the request.
- Directive: Information intended to guide toward the request goal.
- Function: Type of the request such as remedial or preventative treatment.
- ExecutionTime: The expected time required to launch the request.
- Extent: Importance of the change to be made that can be minor, medium, or major.
- Duration: The expected amount of time necessary to complete the request.
- Target: Identification of the entity to be changed by the request.
- Location: Identification of the targeted location by the request.
- PotentialSymptoms: A list of Symptoms that can be indirectly matched as a result of the request process.

Action

A change plan is composed of a sequence of actions that lead to the desired goal. Actions should contains all parameters required by the actuator and may follows a specific order. Before an action can be executed its preconditions must be fulfilled. After the execution, the action yields effects by which the environment changes.

- Parameters: Arguments needed by the actuator to apply this action to a specific target.
- Actuator: Identification of the entity assigned to execute the action.
- Plan: Configuration to which the action belong.
- Order: Arrangement of this action among the separate actions of a plan.
- Target: Identification of the entity to be changed by the action.
- Location: Identification of the targeted location by the action.
- ExecutionTime: Date-time when the action should be executed.
- Duration: The expected amount of time necessary to complete the action.
- PreCondition: what must be established before the action is performed.
- Effect: what is established after the action is performed.

3.5.3 Visualization and administration Interface

FRAMESELF comes with a user friendly visualization interface that enables one to monitor the activity of the autonomic manager main modules. It displays at real time relevant information related to each control loop step. Figure 3.4 shows a snapshot of the visualization interface. We can see the list of discovered sensors and actuators. The input and output of each MAPE-K module related to each management iteration including events, symptoms, RFC, action plans, and actions are displayed as well.

FRAMESELF comes also with a sophisticated administration interface dedicated to system administrator that enable one to configure the autonomic manager, set high level policies, and define required rules. It enables one to display at real time advanced details about the input and out of each MAPE-K components and sub-components. It offers to the administrator the complete picture about each management iteration. Figure3.5 shows a snapshot of the administration interface.

3.6 Home automation self-configuration use case

This use case is selected as a basic example to describe how FRAMESELF can be used as an OM2M service to remotely self-configure a home environment based on knowledge catalog and syntactic rules. Let's consider a home equipped with some legacy devices including a luminosity sensor, a user presence sensor, and a lamp. A home gateway, connected to an external M2M server, is used locally to connect these constrained devices and makes them remotely accessible from outside. The OM2M service platform is used to provide all required service capabilities to ensure communication and data interpretability. The main goal here is to remotely adapt the lamp state according to

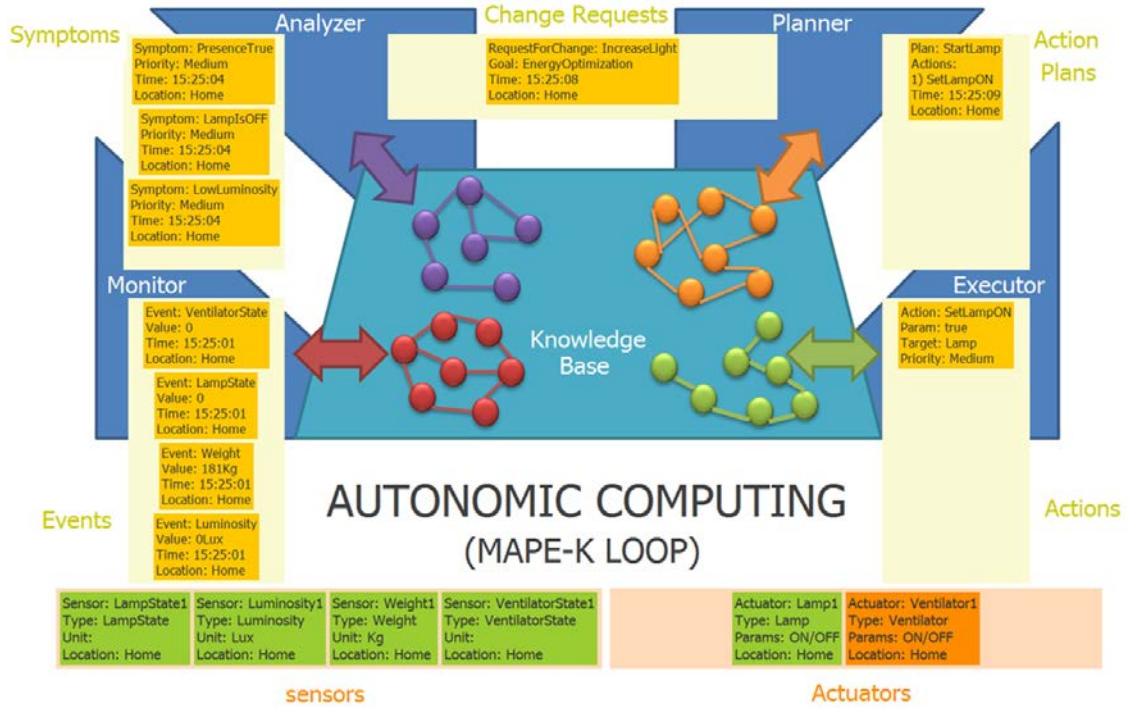

Figure 3.4: FRAMESELF demonstration interface

the luminosity level and the user presence to optimize the energy consumption. The DROOLS reasoning engine and DRL rules [95] are considered for this scenario.

3.6.1 Mapping infrastructure

FRAMESELF is designed to be a generic tool that can be applied to any system by the mean of the mapping infrastructure. Since, the OM2M platform is used, the mapping infrastructure can be implemented simply by using the HTTP or CoAP protocol to interact with the OM2M restful API. In one hand, the specific collector will be implemented as a simple HTTP server that subscribes to the luminosity, user presence and lamp state data container to receive relevant notifications asynchronously. Each received notification will be encapsulated in an Event object and delivered to the monitor module. In the other hand, the specific dispatcher will be implemented as a simple HTTP client to send synchronous requests to change the lamp state. Each received action from the Executer will be converted into an HTTP request. The mapping infrastructure can be implemented using the CoAP protocol as well, since it is supported by OM2M.

3.6.2 Home sensors monitoring

Inside the monitor, the Event_Collector receives event objects collected by the Specific_Event_Collector. The Event_Normalizer can apply several rules to update the received event objects before processing. For example, the following “Luminosity Event Formatting” rule applies a mathematical transfer

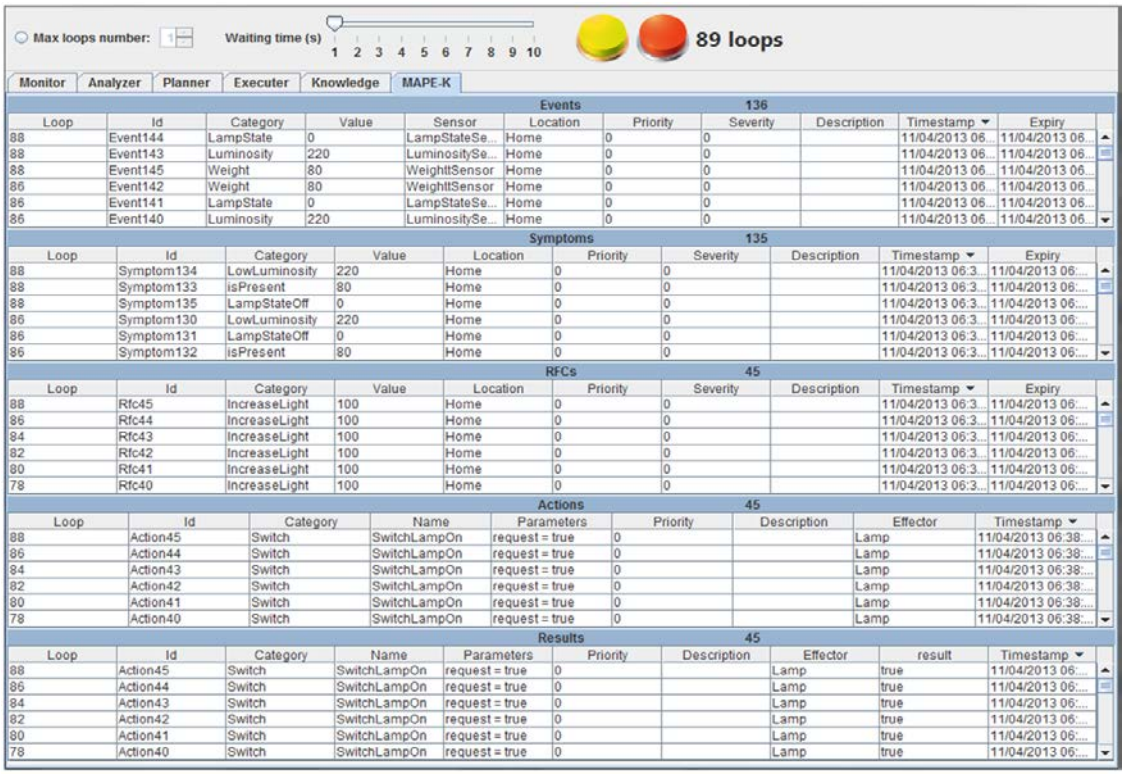

Figure 3.5: FRAMESELF administration interface

function to set the exact value of the luminosity raw value. This rule enables also to set some missing event object fields such as the location, expiration time, etc. in order to homogenize received events before processing.

Event Formatting Rule

```
rule Luminosity_Event_Formatting
    no-loop true
    when
        $event: Event(category == "Luminosity",
            $value: Integer.parseInt(value))
    then
        $event.setValue(($value*0.5)+200);
        $event.setExpirationTime(1235661);
        update($event);
end
```

The Event_Filter applies the following “Expired Events Filtering” rule to compare already formatted events in order to remove out of date ones based on the ExpirationTime field.

Event Filtering Rule

```
rule Expired_Events_Filtering
    when
       Clock($currentDate: date)
       $event : Event(expiry before $currentDate )
    then
       retract($event);
end
```

The Event_Agregator applies the "Duplicated Event Aggregating" rule to compare already filtered events and remove duplicated ones. In our use case, only the most recent events are useful to analyse.

Event Aggregating Rule

```
rule Duplicated_Event_Aggregating
    when
        Event($category: category, $timestamp: timestamp)
        $event: Event(category == $category, timestamp before $timestamp)
    then
        retract($event);
end
```

The Symptom_Inference utilizes the following rules to infer new Symptoms. To make it simple, only three symptom generation rules are presented. The "LowLuminosity Symptom Generation" rule enables one to generate a "LowLuminosity" symptom to alert about the low luminosity level if the luminosity value is less than 300 Lux. Using the same approach, a second rule called "HighLuminosity Symptom Generation" can be defined to generate a "HighLuminosity" symptom when the luminity level is greater than 400 Lux.

LowLuminosity Symptom Generation Rule

```
rule LowLuminosity_Symptom_Generation
    when
        Event(name == "Luminosity", Integer.parseInt(value) < 300)
    then
        Symptom symptom = new Symptom("LowLuminosity");
        insert(symptom);
end
```

The "UserDetected Symptom Generation" rule enables one to generate an "UserDetected" symptom to alert about the presence if the weight sensor detects a value greater than 20kg. A second rule called "UserAbsent Symptom Generation" can be defined to alert about the absence of the user if the weight sensor value is less than 20kg.

UserDetected Symptom Generation Rule

```
rule UserDetected_Symptom_Generation
    when
        Event(name == "Weight", Integer.parseInt(value) > 20)
    then
        Symptom symptom = new Symptom("UserDetected");
        insert(symptom);
end
```

The “LampStateOFF Symptom Generation” rule enables one to generate a “LampStateOFF” symptom to say that the lamp is switched OFF if the lamp state is toggled to 1. Another rule called “LampStateON Symptom Generation” can be used to generate the a “LampStateON” symptom when the lamp state is toggled to 0.

LampStateOFF Symptom Generation Rule

```
rule LampStateOff_Symptom_Generation
    when
        Event(name == "LampState", Integer.parseInt(value) == 0)
    then
        Symptom symptom = new Symptom("LampStateOff");
        insert(symptom);
end
```

The list of generated symptoms are sent to the Analyzer to start to the analyzing step.

3.6.3 Home environment analyzing

The Analyzer applies the “DecreaseLuminosity RFC Generation” rule to generate the following “IncreaseLuminosity” RFC if the luminosity is low, the user is present, and the lamp state is off. It can applies the “DecreaseLuminosity RFC Generation” rule to generate a “DecreaseLuminosity” RFC if the user is present and the luminosity level is low.

IncreaseLuminosity RFC Generation Rule

```
rule DecreaseLuminosity_RFC_Generation
    when
        Symptom(name == "LowLuminosity")
        Symptom(name == "UserDetected")
        Symptom(name == "LampStateOff")
    then
        Rfc rfc = new Rfc("IncreaseLuminosity");
        insert(rfc);
end
```

The list of generated RFCs are sent on the Planner to start the planning process.

3.6.4 New configuration planning

The planner applies the "SwitchLampOn Action Generation" to generate a "SwitchLampOn" action if a "Decrease Light RFC" is received, the lamp actuator is available, and the energy efficiency policy is selected. The "SwitchLampOn" action contains all the parameters required to execute it. The "SwicthLamOff Generation" rule can be applied to generate a "SwicthLampOff" when receiving an "Increase Light RFC". Different action plans can be defined using different policies and addional actuators. The planner can also generate some preventive actions to cope with the side effects that can be generated by RFCs.

SwitchLampOn Action Generation Rule

```
rule SwitchLampOn_Action_Generation
    when
        Rfc(name == "DecreaseLuminosity")
        $effector: Effector(name == "Lamp")
        Policy(name == "EnergyEfficiency")
    then
        Action action = new Action("SwitchLampOn")
        action.addParamter(new Parameter("state","true"));
        insert(action));
end
```

The list of generated actions are set to the Executer for the Execution process.

3.6.5 Home actuators controlling

The Executer uses the Action_Interpreter to verify the coherence of the received sequence of actions. Missing parameters and information related to the execution environment can be added to actions using the Action_Formatter. The Action_Dispatcher sends the the list of formatted actions to the mapping infrastructure. The Specific_Action_Dispatcher take care of converting each action to a HTTP request, sends it to the lamp actuator to change its state. Each returned HTTP response is then converted into a generic action result format and delivered to the executer. The Report_Generator relies on received actions results to generate the control loop execution report report that can be considered in the next iterations.

3.7 Semantic matching and resource reconfiguration

In this section, a more advanced use case will be considered to demonstrate how FRAMESELF solution can be used as an OM2M service to dynamically reconfigure the resource architecture based on an ontology model and semantic rules. In general, an application must perform manually several search operations to discover relevant resources and also several subscriptions to monitor the evolution of interesting resources. In addition, an application has a partial view of the current M2M system. So it becomes very complex to find the right resources especially in a highly distributed M2M system.

Introducing a new autonomic service with a global view of the M2M system capable of configuring the resource architecture when needed is a challenge. The main goal here is to enable any application to dynamically discover interesting devices and to exchange data with the right

communication mode according to its description, role, and relationships. To meet this goal a representative model of M2M system knowledge is required to assist the execution of the control loop steps. Since the M2M ontology covers all the M2M concepts required for this scenario, it will be used as a knowledge model within FRAMESELF knowledge base.

For simplicity, figure 3.6 shows a partial view of the M2M ontology highlighting the main concepts needed for the self-configuration process. In one hand, the M2M ontology enables one to represent physical things like sensor and actuator devices that respectively observes and acts on a quantity kind. In the other hand, it enables one to represent non physical things like monitor and controller managers than respectively monitors and controls a quantity kind. Each thing is registered to one node and offers a set of services. Additional concepts of the M2M ontology can be considered to further refine the semantic matching process for a more advanced configuration.

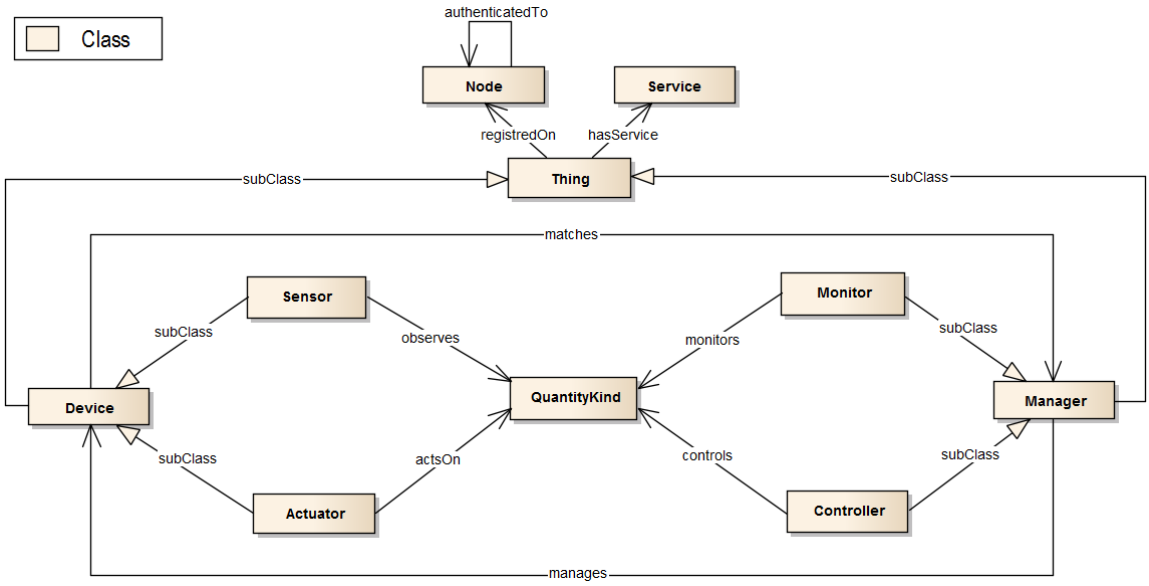

Figure 3.6: Ontology model for M2M

We consider a building equipped with a gateway connecting locally an electricity meter and a lamp that created on the home gateway GSCL respectively an "ElectricitySensor" and "LampActuator" applications. A lamp controlling application "LampController" is registered on the user Smartphone DSCL to enable the user to manually control his lamp. In addition, a smart metering application "ElectricityMonitor" is registered on the smart metering server NSCL to collect the building consumption. FRAMESELF should discover and monitor the description of all registered applications, reasons on the M2M ontology model using semantic rules to find relevant matching, and finally reconfigure the resource architecture accordingly to set up the required communications.

3.7.1 Mapping infrastructure

Since OM2M offers a restful API to manage resources, the mapping infrastructure can be implemented simply using a HTTP server as Specific_Event_Collector, and a HTTP client as a Spe-

cific_Action_Dispatcher. A CoAP implementation can be used as well. In this use case FRAMESELF will focus on the meta-data of the applications and not the measured data contrary to the previous example. On the monitor side, the Specific_Event_Collector will subscribe to the applications descriptor container to monitor their description. Each received notification will be encapsulated in an Event object and delivered to the monitor module. On the Executer side, a sequence of actions will be delivered to the Specific_Action_Dispatcher. Each action is converted into an HTTP request and sent to the right SCL to reconfigure it.

3.7.2 M2M resources discovery

Initially, the monitor tries to find all existing nodes within the M2M system. It retrieves the "sclBase" resource of the NSCL and so adds the "Server" individual as an instance of the NSCL class in the M2M ontology instance. It searches also for the authenticated D/GSCLs and so adds the "Gateway" individual as an instance of GSCL class, and the "Smartphone" individual as an instance of the DSCL class. Then, it subscribes to the "scls" collections of the NSCL to be notified when new D/GSCLs are authenticated. Then, for each discovered node, the monitor retrieves the registered applications and the offered services and operations, and adds them to the ontology instance. Figure 3.7 shows a snapshot of the corresponding M2M ontology instance as generated by the FRAMESELF monitor.

- The "ElectricitySensor" individual is added as an instance of the Sensor class. It is registered to the "Gateway" GSCL, and is linked to the "Electriciy" quantity kind using the "observes" relationship. This sensor provides a service called "ElectricityService" that offers several operations to collect data from the sensor. In particular, the operation "SubscribeToElectricitySensorOperation" describes the address, method, and content input needed to subscribe monitors to this sensor.

- The "LampActuator" individual is added as an instance of the Actuator class. It is registered also to the "Gateway" GSCL, and is linked to the "Luminosity" quantity Kind using the "actsOn" relationship. This actuator provides a service called "LampService" that offers several operations to command the actuator. Specially, the operation "LampCommandOperation" describes the address, method, and content input needed to command the lamp actuator.

- The "LampController" individual is added as an instance of the Controller class. It is registered to the "Smartphone" DSCL, and is linked to the "Luminosity" quantity Kind using the "controls" relationship. This controller provides a service called "LampControllerService" that offers several operations to use the controller. In particular, the operation "SubscribeToLampOperation" describes the address, method, and content input needed to subscribe actuators to this Controller.

- The "ElectricityMonitor" individual is added as an instance of the Monitor class. It is registered to the "Server" NSCL, and is linked to the "Electricity" quantity Kind using the "monitors" relationship. This monitor provides a service called "ElectricityMonitorService" that offers several operations to use the monitor. In particular, the operation "MonitoredDataRetrieveOperation" describes the address, method, and content input needed to fetch the collected electricity data.

Any relevant event detected on the M2M system architecture like the authentication of a new node, the registration of a new application, the delete of an existing application, or the update on any application description is transformed to a symptom and then sent to the Analyzer for verification.

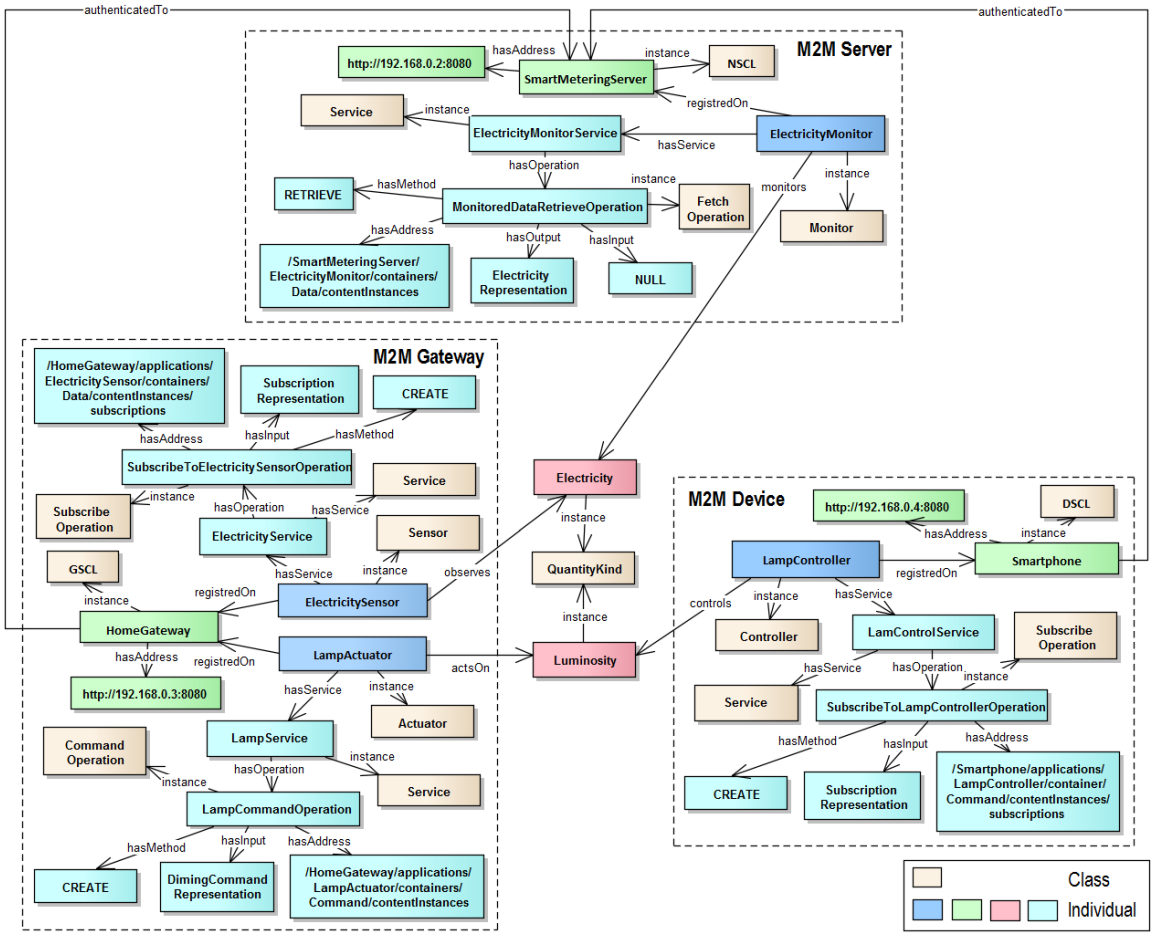

Figure 3.7: M2M ontology instance

3.7.3 Application and device semantic matching

The Analyzer aims to analyze the current M2M architecture in order to find semantic matching between registered applications based on their description. Inference rules can be applied to infer new knowledge and so enrich the current M2M ontology instance with new individuals and relationships. This new knowledge is necessary to understand the role of each thing in the M2M architecture. It allows each thing to take maximum advantage of the services offered by other things. In this example, two SPARQL rules are applied by the analyzer to find semantic matching between the resisted managers and devices. The first rule “Matching sensors to monitors” says: if there is a

sensor that observes a particular quantity kind, and there exists also a monitor that monitors the same quantity kind, and if this sensor is not already matched to that monitor, then as a result a new "matches" relationship is inferred to link together this sensor and monitor.

Matching sensors to monitors

```
CONSTRUCT {
    ?sensor m2m:matches ?monitor
}
WHERE {
    ?monitor rdf:type  m2m:Monitor .
    ?sensor rdf:type  m2m:Sensor .
    ?qKind rdf:type  m2m:QuantityKind .
    ?monitor m2m:monitors  ?qKind .
    ?sensor m2m:observes  ?qKind .

    FILTER EXISTS { ?sensor m2m:matches ?monitor }
}
```

The second rule "Matching actuators to controllers" says that if there is an actuator that acts on a particular quantity kind, and there exists also a controller that controls the same quantity kind, and if this actuator is not already matched with that controller, then a new "matches" relationship is inferred to link together this actuator and controller.

Matching actuators to controllers

```
CONSTRUCT {
    ?actuator m2m:matches ?controller
}
WHERE{
    ?controller rdf:type  m2m:Controller .
    ?actutator rdf:type  m2m:Actuator .
    ?qKind rdf:type  m2m:QuantityKind .
    ?controller m2m:controls  ?qKind .
    ?actuator m2m:actsOn  ?qKind .

    FILTER EXISTS { ?actuator m2m:matches ?controller }
}
```

Using the same approach, a more advanced rules can be applied setting more constraints such as the device location, temporal requirements, or quality of services parameters. The list of inferred links are transformed to RFCs by the analyzer and sent to the planner.

3.7.4 New connections planning

The planner is responsible of elaborating the required action plans to interconnected matched applications. For each device matching a manager application, the planner checks the provided services, then looks for the appropriate operations to trigger to establish the communication.

The following SPARQL rule enables one to retrieve, for each sensor/monitor matching, four relevant fields required to generate the subscription request. These fields are extracted from the op-

eration "SubscribeOperation" of the sensor and also the operation "FetchOperation" of the monitor as follows:

- "sensorAddress": contains the address of the Sensor container where the subscription should be done. In our use case, it includes the address of the "ElectricitySensor" data container "/Gateway/applications/ElectricitySensor/containers/Data/contentInstances/subscriptions",
- "method": contains the subscription request method (CREATE),
- "input": contains the subscription representation type (SubscriptionRepresentation),
- "monitorAddress": contains the address of the Monitor container to be used as contact URI. In our use case it is the address of the "ElectricityMonitor" data container "/Server/applications/ElectricityMonitor/containers/Data/contentInstances".

Retrieve Monitor-to-Sensor subscription request parameters

```
SELECT ?sensorAddress ?method ?input ?monitorAddress
WHERE{
    ?sensor m2m:matches ?monitor .

    ?sensor m2m:hasService ?sensorService .
    ?sensorService m2m:hasOperation ?sensorOperation
    ?sensorOperation rdf:type ?SubscribeOperation .
    ?sensorOperation rdf:hasAddress ?sensorAddress .
    ?sensorOperation rdf:hasMethod ?method .
    ?sensorOperation rdf:hasInput ?input .

    ?monitor m2m:hasService ?monitorService .
    ?monitorService m2m:hasOperation ?monitorOperation .
    ?monitorOperation rdf:type ?FetchOperation .
    ?monitorOperation rdf:hasAddress ?monitorAddress .

    FILTER EXISTS { ?monitor iot:manages ?sensor }
}
```

The following SPARQL rule enables one to retrieve, for each actuator/controller matching, four relevant fields required to generate the subscription request. These fields are extracted from the operation "CommandOperation" of the actuator and also the operation "SubscribeOperation" of the controller as follows:

- "controllerAddress": includes the address of the Controller container where the subscription should be done. In our use case, it is the address of the "LampController" command container "/Smartphone/applications/LampController/containers/Command/contentInstances/subscriptions",
- "method": contains the subscription request method (CREATE),
- "input": contains the subscription representation type (SubscriptionRepresentation),

- "actuatorAddress": includes the address of the Actuator container to be used as contact URI. In our use case, it includes the address of the "LampActuator" command container "/Gateway/applications/LampActuator/containers/Command/contentInstances".

Retrieve Actuator-to-Controller subscription request parameters

```
SELECT ?controllerAddress ?method ?input ?actuatorAddress
WHERE{
    ?actuator m2m:matches ?controller .

    ?actuator m2m:hasService ?actuatorService .
    ?actuatorService m2m:hasOperation ?actuatorOperation
    ?actuatorOperation rdf:type ?CommandOperation .
    ?actuatorOperation rdf:hasAddress ?actuatorAddress .


    ?controller m2m:hasService ?controllerService .
    ?controllerService m2m:hasOperation ?controllerOperation .
    ?controllerOperation rdf:type ?SubscribeOperation .
    ?controllerOperation rdf:hasAddress ?controllerAddress .
    ?controllerOperation rdf:hasMethod ?method .
    ?controllerOperation rdf:hasInput ?input .

    FILTER EXISTS { ?controller m2m:manages ?actuator }
}
```

For each matching, the four retrieved fields are converted into an action, then sent to the Executor module. In our scenario two actions will be generated. The first one "ElectricitySubscriptioAction" aims to subscribe the "ElectricityMonitor" to the "ElectricitySensor". The second one "LampSubscriptionAction" aims to subscribe the "LampActuator" to the "LampController".

3.7.5 Resource dynamic reconfiguration

The Executor aims to execute the planned actions through the HTTP client, and to report the execution result into the knowledge base. Each received Action is converted into an HTTP request. Each HTTP response is converted into an action result.

The following snippet shows the result of the conversion of the action "ElectricitySubscriptioAction" to a HTTP request. The "Create" method is converted into a "POST" method, the input field is converted into a subscription XML representation. The "ElectricitySensor" "Data" container address is entered as HTTP Request-URI, the "ElectricityMonitor" "Data" container address is defined in the "contact" tag of the subscription XML representation.

Electricity subscription HTTP request

```
POST /Gateway/applications/ElectricitySensor/containers/Data/
     contentInstances/subscriptions HTTP/1.1
Host: www.server.com

<om2m:subscription>
   <om2m:contact>
        /Server/applications/ElectricityMonitor/containers/Data/contentInstances
   </om2m:contact>
</om2m:subscription>
```

The following snippet shows the result of the conversion of the action “LampSubscriptionAction” to a HTTP request. The “Create” method is converted into a “POST” method, the input field is converted into a subscription XML representation. The “LampController” “Command” container address is entered as HTTP Request-URI, and the “LampActuator” “Command” container address is defined in the “contact” tag of the subscription XML representation.

Lamp subscription HTTP request”

```
POST /Gateway/applications/ElectricitySensor/containers/Command/
     contentInstances/subscriptions HTTP/1.1
Host: www.server.com

<om2m:subscription>
   <om2m:contact>
      /Gateway/applications/LampActuator/containers/Command/contentInstances
   </om2m:contact>
</om2m:subscription>
```

To make it simple, the NSCL node address “www.server.com” is defined as Host HTTP header for all HTTP requests. Therefore, the executor will send these requests directly to the NSCL. Thanks to the re-targeting mechanism, these HTTP requests will be redirected automatically to the appropriate D/GSCL according the HTTP Request-URI.

The “Electricity Subscription HTTP request” will be redirected to the Gateway. Once the subscription is created on the “ElectricitySensor”, the “ElecricityMonitor” will automatically receive all the electricity measurement. The “Lamp Subscription HTTP request” will be redirected to the Smartphone. Once the subscription is created on the “LampController”, the “LampActuator” will automatically receive all the lamp command.

If a HTTP request is executed successfully with a “201 Created” status code, then the executor adds the “manages” relationship between the corresponding manager and device. Otherwise if a bad status code is returned, then an alert including the error details is reported. As soon as all actions are executed and reported, a new control loop can start.

3.8 Conclusion

In this chapter, we proposed the FRAMESELF framework to extend the OM2M platform services with self-management capabilities. FRAMESELF provides a detailed architecture with a complete data model and visualization interfaces for autonomic based on the MAPE-K control loop reference model. Its features and capabilities was demonstrated first using a self-optimization home automation scenario. Second, a more advanced scenario focusing on the self-configuration of the M2M resource architecture is illustrated making use of the IoT-O ontology and semantic reasoning. OM2M, IoT-O and FRAMESELF constitute together an efficient solution to reduce M2M system complexity and get closer to effective horizontality.

In the next chapter, we explain why the current internet infrastructure cannot support mass-scale deployment of M2M solutions. As an alternative, we propose to extend OM2M with new information centric networking and data dissemination mechanisms to go towards M2M scalability.

Chapter 4

OSCL: Information centric M2M network

Contents

4.1 Introduction 87
4.2 Network overlay in M2M systems 88
4.3 Overlay Service Capability Layer 91
4.4 OSCL example use cases 93
4.5 OSCLsim: NDN simulator for OSCL 95
4.6 Dynamic random graph model for diameter constrained networks . 99
4.7 DRG model simulations 103
4.8 DRG model example applications 104
4.9 Conclusion 109

4.1 Introduction

M2M systems rely in general on a centralized communication model in which the M2M server is responsible of handling and relaying all distributed signaling in the system. It is well known that centralized systems can surely yield strong optimization but, at the same time, critical entities of the system that act as concentrators may suddenly become single points of failure, thus lowering the fault tolerance of the entire system. Furthermore, even if the M2M server can be implemented in a distributed and elastic way on the cloud, the M2M system scale and costs would be much higher if the capabilities of distributed gateways and devices are not fully exploited. In return, more advanced techniques should be used for network sizing to enhance system scalability.

In this chapter we present a state of the art of network overlay approaches and mechanisms that can be used in M2M systems with a focus on Information Centric Networking (ICN) [32] alternative. We provide an overview of the Named Data Networking (NDN) [33] architecture and show how it can enhance OM2M networking and data dissemination. As a solution we present a meshed Overlay of Service Capability Layers (OSCL) [34], designed in accordance to NDN architecture.

Main capabilities are illustrated using use case examples. Then, we describe our OSCLsim, an NDN simulator developed specifically to experiment the performance of the proposed approach and highlight the evaluation results. Unfortunately, in many real systems, the properties of their interacting units cannot be deterministically known in advance. As a solution, we present a dynamic random graph (DRG) [35] model for diameter constrained topologies. We show different applications for measuring some OSCL properties such as rank bound, robustness, transient duration and link failures.

The two main contributions of this chapter [34, 35] are conducted in collaboration with Dr. Alfredo Grieco, associate Professor in telecommunications at Polytechnic University of Bari.

4.2 Network overlay in M2M systems

The use of network overlays in M2M systems has been widely studied in literature with reference to many different facets. A summary of the different problems faced so far along with corresponding approaches are reported in the sequel.

4.2.1 Wireless Mobile and Sensor Networks

The proposals formulated in [26–28] mainly target performance optimization in wireless mobile and sensor networks in order to speed up data transfer and lower energy consumption. In [28], it is proposed to exploit Mobile Ad-hoc NETwork (MANET) overlays, autonomously and collaboratively formed over WSNs, to boost urban data harvesting in IoT. Such overlays are used to dynamically differentiate and speed up the delivery of urgent sensed data made available using the latest emergent standards/specifications for WSN data collection. In the context of mobile systems, in [27] it is proposed to map virtual clones of M2M nodes onto an overlay using a Cloud in order to let virtual clones interacting to each others thus minimizing the energy spent by wireless devices. In [26] application level overlays are used to fasten the dissemination of urgent data in a WSN based on forest based topologies.

4.2.2 Distributed Hash table (DHT)

On the other side, approaches conceived in [29, 30] are mainly devoted to resource discovery using structured overlays of proxy servers. In [29], it is proposed to build an overlay of proxy servers in a M2M deployment in order to handle resource discovery through P2P mechanisms. In particular, the location of a target resource is obtained by querying an overlay based on the eXtendible Metadata Hash Table (XMHT) [30], which is an extension of the widely used Pastry protocol [96]. This approach in itself is very valuable and deserves further investigations to understand to what extent it can fit complex M2M scenarios in which the level of popularity of different resources is highly heterogeneous. In fact, on one hand, a Distributed Hash Table (DHT), which is the main element of this system, can be used to effectively locate rare items because its key-based routing is scalable. On the other hand, it may incur significantly higher overheads than unstructured approaches for popular content [97]. Also, at the present stage, a debate is still open about the effectiveness of DHT in very large scale systems [98, 99].

4.2.3 Host Identity Protocol (HIP)

Another branch of literature is mainly devoted at conceiving extensions to the well known Host Identity Protocol (HIP) [31], which was originally proposed to handle Internet host identities and to decouple the host name space from the network address name space. In particular, a HIP-based M2M overlay network named HBMON is proposed in [100]. It extends the standard HIP in several directions in order to define, organize and manage device memberships within the overlay. Also HBMON proposes a novel IPv6 address assigning method in order to configure the overlay members with private IPv6 addresses. This architecture has been further improved in [101] and [102] in order to embrace mobility and autonomic aspects too.

4.2.4 eXtensible Messaging and Presence Protocol (XMPP)

The idea of using the eXtensible Messaging and Presence Protocol (XMPP) to discover entities with a fully distributed approach in an M2M overlay has been discussed in [103]. This service discovery protocol works at the application layer inside an XMPP client implementation. It can be categorized as an unicast protocol, as it sends direct discovery messages to known receivers. Its purpose is to discover M2M overlay entities presented as Jabber Identifiers (JIDs). In [104] another interesting P2P approach to M2M residential home applications has been investigated too. These two latter approaches, albeit promising, has been conceived for dealing with simple M2M deployments and its effectiveness in large scale M2M systems deserves further analysis.

4.2.5 Enhanced Dynamic Service Overlay Network (e-DSON)

The enhanced Dynamic Service Overlay Network (e-DSON) is proposed in [105] to handle service composition in context aware and user centric M2M applications. Basically the e-DSON could serve to orchestrate the usage of the data made available by a SmartM2M system in order to allow the deployment of new services.

4.2.6 Information Centric Networking (ICN)

Host-centric communication principles rule the Internet behavior and are at the basis of most standard applications (Web, Email, FTP, etc) mainly relying on well defined servers and hosts. However, the way users perceive network services has undergone through a substantial evolution during last years [106]. Nowadays, users ask for contents, regardless the physical location of the server, thus requiring novel networking primitives beyond the Internet Protocol (IP). In addition, the currect Internet suffers of a number of problems related to: (i) service provisioning across different wireless access technologies (i.e., mobility); (ii) security issues; (iii) environment with sporadic connectivity; (iv) network and server congestion, all deriving from the host -centric rationale.

The so-called Information Centric Networking (ICN) promises to overtake these issues by posing contents at the core of networking primitives. In other words, according to the ICN paradigm, packet switching rules should be based on the content to retrieve rather than the host address from which the content could be downloaded.

The idea of incorporating data awareness in M2M systems is being actively investigated and several ICN based proposals have been formulated [107–113]. The pioneering study in [107] demonstrated that attribute-named data at the lowest levels of communication can improve the energy efficiency of WSNs thanks to in-network processing with filters, data aggregation, and nested queries.

Later on, when ICN architectures started being investigated worldwide, the idea to approach M2M systems using data centric networks lead to different interesting solutions and opened many implications [110]. Noticeable contributions in [108] and [109] demonstrated the benefits arising from using an ICN approach in M2M systems, based on the the well known CCN [106] and Publish Subscribe Internet Technology (PURSUIT) architectures, respectively. More specific applications of CCN in home networks have been presented in [111–113].

4.2.7 Named Data Networking (NDN)

The NDN [33] architecture is based on the Content Centric Networking (CCN) rationale [106] that assumes hierarchical content names, receiver initiated sessions, content level security schema, and in-network caching mechanisms. The NDN architecture is born to natively support mobile applications and multi-cast data dissemination while, at the same time, to improve content distribution services across heterogeneous platforms.

The entire NDN paradigm is based on two packets: *Interest* and *Data*. When a consumer of data needs a given information it just sends an *Interest* message in which it is specified the hierarchical name of the requested item. This *Interest* message is routed towards one or multiple nodes that are in possess of the requested content, which is then encapsulated in a *Data* message and sent back as an answer to the consumer.

Along its path to the requester, the *Data* message can be stored in the cache of relaying nodes, that will become able to answer to future requests for the same content, thus lowering the load on the servers that store permanent copies of the asked item. Furthermore, if an NDN router receives multiple requests for the same content that have been issued by different nodes, it will forward only the first *Interest* packet and it will suppress next ones by keeping note of their arrival faces, thus enabling a native multi-cast dissemination model.

Accordingly, three data structures are used in NDN: the Content Store (CS), representing the cache memory; the Forwarding Information Base (FIB) that plays the same role as in classic IP routers; and the Pending Interest table (PIT) in which it is possible to keep track of the arrival faces of the *Interest* messages received in the past and for which a *Data* packet has not been yet received.

Interest-based routing is one of the most important aspects in NDN from the perspective of this contribution. In fact, as it will be shown later in Sec. 4.3, the OSCL proposed in this work strongly depends upon the capability of locating at least one copy of a given content in NDN, knowing its hierarchical name. To this end, it is worth remarking that in NDN the strategy layer is responsible for routing *Interest* messages and that several algorithms can be used for that purpose [114]. For example, [115] proposes to build routes towards the permanent copies of contents through announcements. On the other hand, other proposals target mechanisms for locating the cached copies of requested items. In this case, when the shortest path towards a given item is not known a priori, random routing and flooding represent the most simple solutions to use, but they might incur high response time and overhead, respectively [116]. On the other hand, different strategies based on Machine Learning techniques [117] andor Bloom Filters [116, 118–120] could be adopted to route Interest messages in an effective way, based on the past history and the exchange of routing information among neighboring NDN nodes.

4.3 Overlay Service Capability Layer

The SmartM2M standard proposes a powerful architecture to break the barriers due to the heterogeneity of M2M devices and systems. Unfortunately, it is based on a very centralized communication model that concentrates at the NSCL many operations that could be handled in a distributed way. We propose the concept of OSCL to increase scalability, robustness, and flexibility of SmartM2M systems. From a networking perspective, the OSCL is a meshed overlay that inter-connects many SCL instances to allow the provisioning of M2M services without the compulsory inter-mediation of a centralized NSCL.

In the SmartM2M standard, all DSCLs and GSCLs should be mutually authenticated to the NSCL. In absence of mobility, this operation is executed only once per device and allows the M2M system to handle future resource discovery operations. This kind of discovery is centralized and it uses the NSCL as a relaying point of all signaling messages. On the opposite, in our proposal, it will be used as a fallback solution only when the OSCL fails to handle a discovery operation.

Similarly, after a resource has been discovered, the NSCL is in charge to relay subscriptions to that resource (e.g., the sensing service of an electricity metering application) and all the data sent to subscribers (e.g., a remote energy monitoring application). Contrariwise, we propose to leverage OSCL capabilities to access to resources in a distributed way., i.e, without the compulsory inter-mediation of the NSCL. In this way, as soon as new service discovery operations are executed it becomes possible to subscribe to a target resource and to exchange data by means of a meshed overlay network that inter-connects remote SCL instances.

It is also worth remarking that the schema proposed hereby requires sophisticated approaches to locate a resource within a meshed overlay, which adopts in network caching techniques and whose structure cannot be known a priori. A promising solution to this problem can be formulated according to an Information Centric Networking (ICN) approach which enables name based content discovery and networking mechanisms.

In the SmartM2M standard, the resources are named according to a hierarchical scheme. Accordingly, we propose to adopt the NDN architecture to handle the services offered by the OSCL. Thanks to this choice a named resource can be located by simply issuing an *Interest* message containing its name. This message will be routed within the OSCL under the control of the NDN strategy layer and it will trigger at least one *Data* message containing the locator of the target resource. Many routing algorithms are nowadays available for NDN based on different kind of assumptions [32].

The choice is application specific and can be left open in order to allow customized M2M deployments. It is worth to remark that *Interest* routing operations could fail for many reasons (e.g., stale information in the Forwarding Information Base (FIB) of the nodes, resource not reachable due to a not connected topology graph, research scope too limited to reach the target resource). Under such circumstances, the classic centralized resource discovery already available in the SmartM2M standard is used as a backup solution.

After the resource discovery has been accomplished, a *QoS Monitoring* phase is executed. If the resource was discovered using the distributed approach, then a path in the OSCL already exists that could be used for subsequent subscriptions and data exchange phases. The *QoS Monitoring* phase will measure the key performance indicators of the path (e.g., packet loss ratio, throughput, delay) and if the quality of the path will result to be poor a new link will be established in the OSCL to enable a direct access to the target resource. The link will be set up using the locator information provided by the discovery phase. Of course, if the distributed discovery failed and, as a consequence,

the centralized one was used, a new link will be immediately set up in the OSCL. We remark that, similarly to [121], we extend the *Interest* header, and related in network handling operations, in order to solicit the transmission of more than one *Data* message by the target resource.

In summary, the OSCL enables the following enhancements:

Distributed Service Discovery. It is used to discover a resource in the M2M system based on ICN primitives. This phase, if successful, opens at least one path in the OSCL towards the discovered resource. Also, it returns both the URI of the resource and its network locator, e.g., IP address and Port. These two additional information are useful to handle alias and to establish a new link in the OSCL (if required), respectively.

P2P Subscription & Data Exchange. Thanks to the discovery phase, subscriptions to a target resource and subsequent exchange of information can be handled without the centralized control of the NSCL, i.e., a multihop path in the OSCL can be used.

QoS Monitoring. The path activated using the *Distributed Service Discovery* is composed by a sequence of logical links and nodes within the OSCL. It can be used to access to the remote resource. At the same time, the QoS this path is able to provide could not be enough for effectively supporting *P2P Subscription & Data Exchange* phases. As a matter of fact, some nodes within the overlay could not be able to effectively act as network relayers and thus worsening the QoS due to additional delays and packet losses. For this reason, the OSCL we envisage also include a *Distributed QoS Monitoring* service that establish a new link with the remote resource when the path being used is not suitable anymore for a given M2M application.

A simplified flow chart of all afore discussed phases is reported in Fig. 4.1.

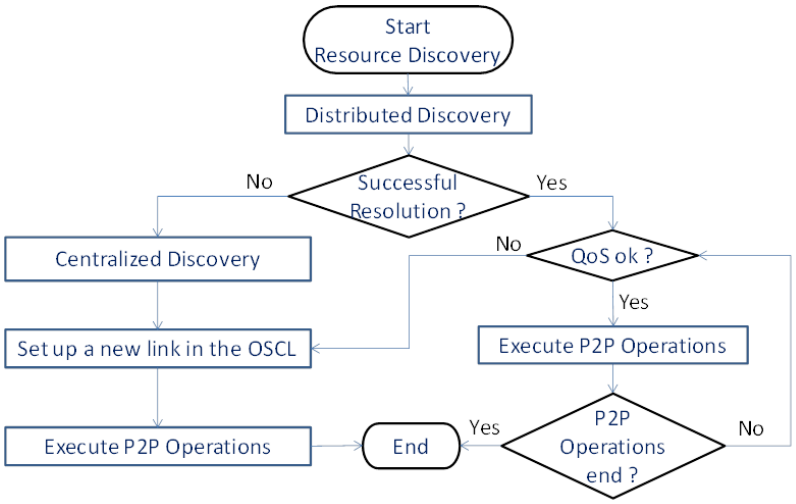

Figure 4.1: OSCL operations: simplified flow chart.

Notice that all these new features are optional so that the legacy interoperability with existing SmartM2M deployments would be ensured together with the possibility of a graceful upgrade of currently available systems.

4.4 OSCL example use cases

In this section we describe two basic uses cases to demonstrate the main features of the proposed approach. The first one is made of the M2M Network, one gateway, and one device (see Fig. 4.2). The gateway and the device host an electricity metering application and an energy monitoring application, respectively. The monitoring application needs to read the data sensed by the metering process. Before this becomes possible, if OSCL is not used, it is necessary to accomplish the following actions:

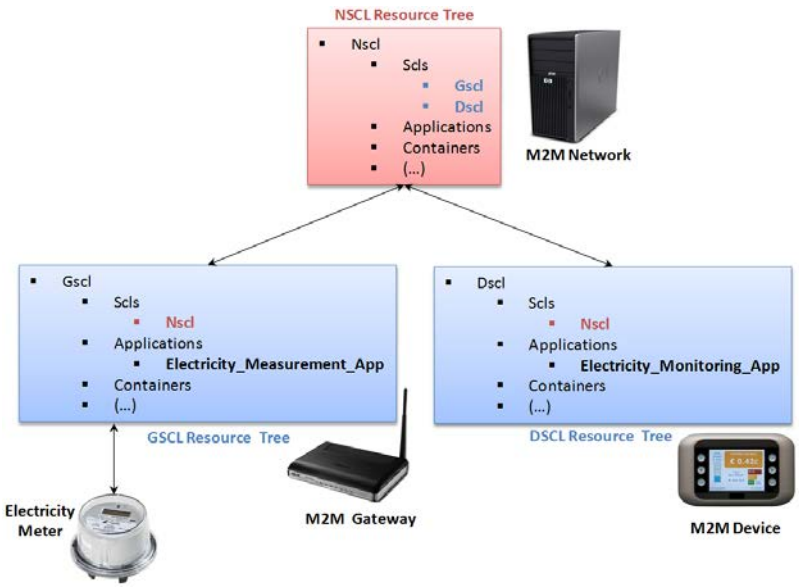

Figure 4.2: A Simple ETSI M2M scenario.

1. the gateway and the device mutually authenticate to the NSCL.
2. the electricity metering application registers to the GSCL and then creates a data container.
3. the monitoring application registers to the DSCL.
4. the monitoring application discovers the name of the gateway by querying the NSCL and then discovers the name of the application by querying the GSCL. Both queries are routed by the NSCL. At the end of this process the monitoring application knows the URI of the metering application (e.g., GSCL/applications/electricity_meter).
5. the monitoring application issues a subscription targeted to that URI. This subscription is routed by the NSCL towards the GSCL. after the GSCL accepts the subscription, every time the electricity metering application pushes a new event to the data container (i.e., a new measurement) a notification is sent to the monitoring application (through the NSCL).

By applying the OSCL to this use case, it becomes possible to increase the scalability of the SmartM2M system. In fact, as shown in Fig. 4.3, the NSCL is no longer in charge of relaying the

data sensed from the metering application. Straight lines represent the SmartM2M messages used also by the OSCL. Dashed lines represent the new messages introduced with the OSCL. The gray lines are the SmartM2M messages that are no longer used thanks to the OSCL.

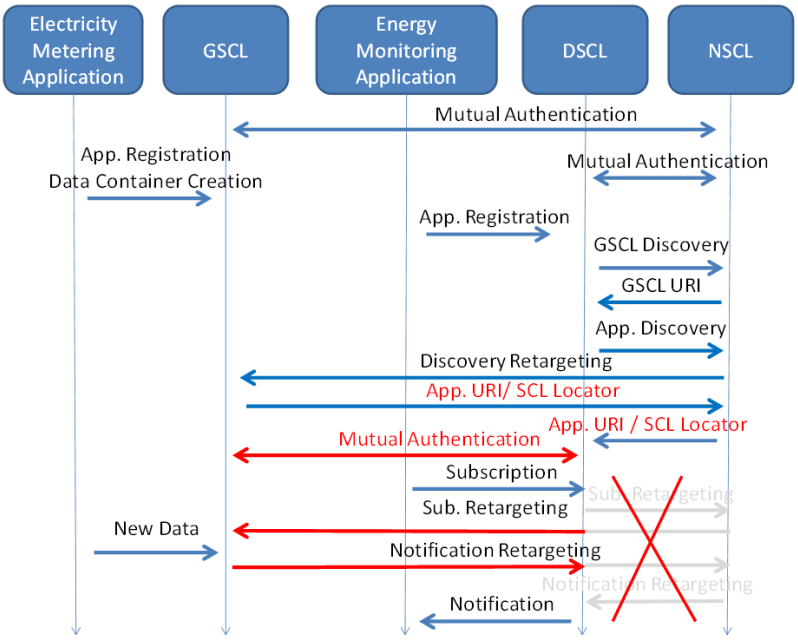

Figure 4.3: Message Sequence Chart with OSCL.

Also, from the message sequence chart in Fig. 4.3, it is possible to note as the authentication, discovery, subscription, and data exchange operations do not require significant modifications to SmartM2M messages: they only entail different endpoints in the exchange of packets.

A slightly more complex scenario is depicted in Fig. 4.4, where two more gateways are present with respect to the previous case. This second scenario allows us to detail how NDN messages (i.e., *Interest* and *Data*) are routed through multiple GSCL instances in the OSCL.

Fig. 4.5 shows that the *Interest* message opens a path towards the target resource during the discovery phase. This path is then used to transport both signaling and application data (encapsulated in *Data* messages) between two remote SCL instances in the OSCL, without any mandatory intervention of the NSCL. It is also worth to note that relaying data through a multi-hop path reduces the number of logical links established in the overlay per each SCL instance thus further increasing the scalability of the system.

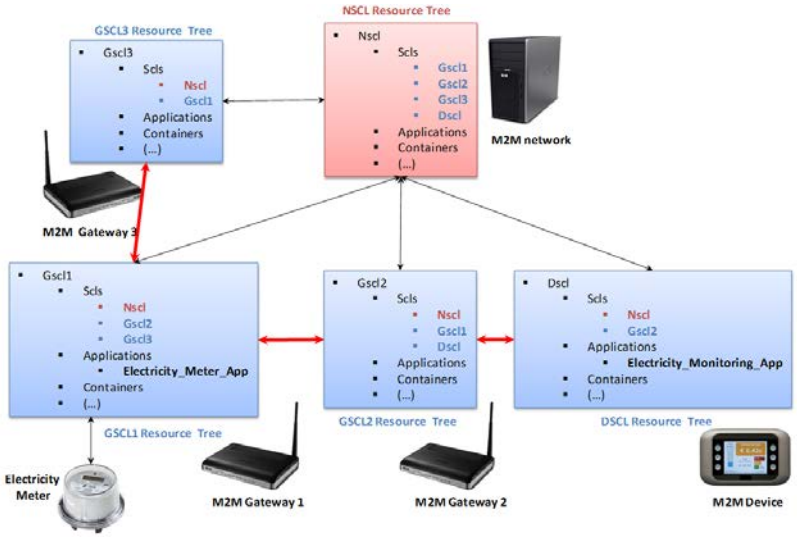

Figure 4.4: An extended SmartM2M scenario with P2P capabilities.

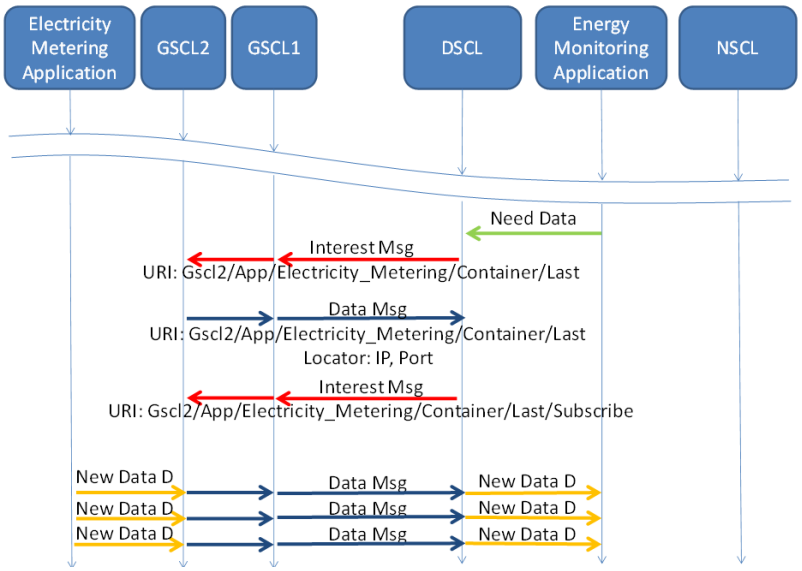

Figure 4.5: Message Sequence Chart with OSCL and ICN messages.

4.5 OSCLsim: NDN simulator for OSCL

In this section we present the OSCLsim, an OSCL simulator designed and developed to validate the proposed approach.

4.5.1 OSCLsim features and interface

Fig. 4.6 shows the graphical interface of OSCLsim which is composed of four main parts. Part (1) illustrates the network graph behavior and enables one to follow at real time the creation of new links, the loss of links, the selected multi-hop path for exchanging data, and the position of cached data. Part (2) enables one to follow the distribution of the performed transactions within the system nodes. Part (3) enables one to follow the evolution of the number of established links in the network. Part (4) enables one to configure various simulation parameters such as the number of nodes, the maximum number of hop, the links failure rate, the number of unique data, the simulation duration, the simulation speed, etc.

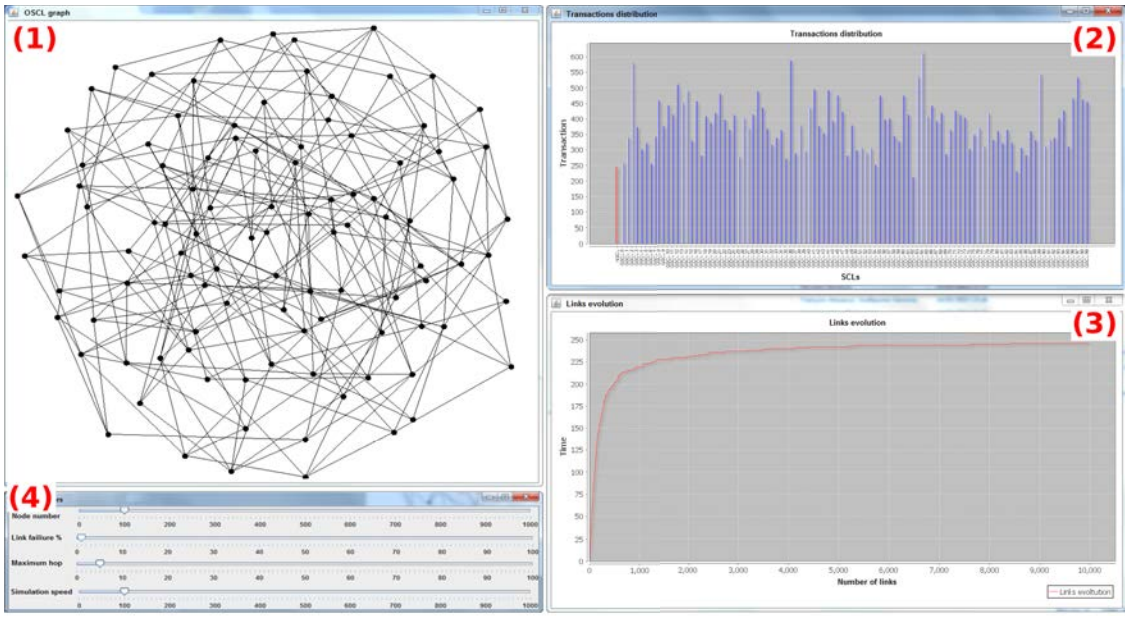

Figure 4.6: SmartM2M NDN simulator graphical interface

To provide a further insight, Fig. 4.7 pictures the evolution of an OSCL graph with 1000 nodes and 5 maximum hops. (a) shows the graph at the beginning of the simulation, when no edge is present. (b) shows the graph at the transient phase, when a few edges have been created and a new edge is added. (c) shows the graph at steady state, when all required paths (no longer than 5 edges) have been already created. An existing path composed of 5 hops is selected to exchange the data.

4.5.2 OSCL performance evaluation

To check the OSCL performance, let's consider an M2M system composed of 1 NSCL and 100 GSCL. The number of request is set to 1000, the maximum number of hops is set to 5, the link failure rate is set to 0, and the number of unique data is set to 100. Three different scenarios are

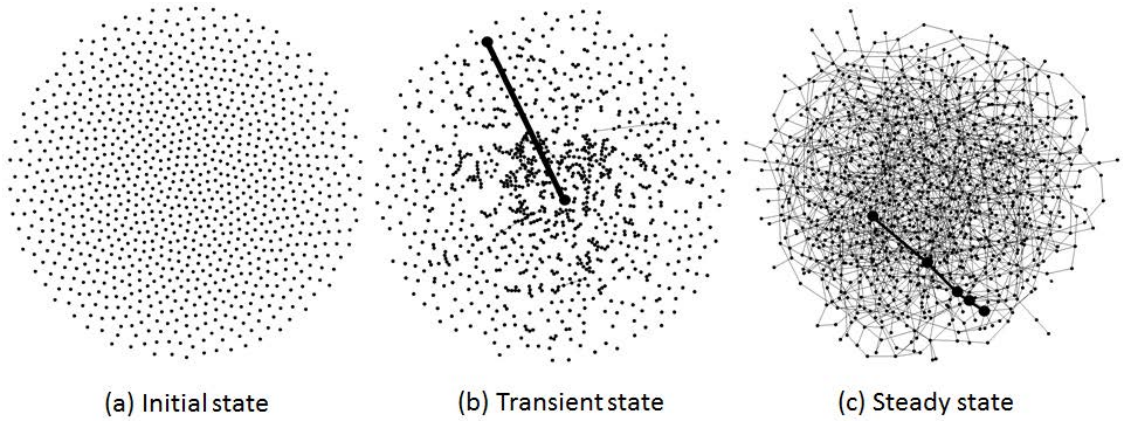

Figure 4.7: Evolution of the OSCL network

performed to check the advantage of OSCL compared to the centralized approach and also to check the advantage of caching facing scapability

The first simulation is performed in a traditional SmartM2M system. In other terms the OSCL and data caching are deactivated. Fig. 4.8 shows that the total number of NSCL transactions is about 880 and no transactions are done by the GSCLs. The number of transactions is less than the number of requests because it may happen that the requester and the requested data are hosted in the same node so no re-targeting transactions are required in this case.

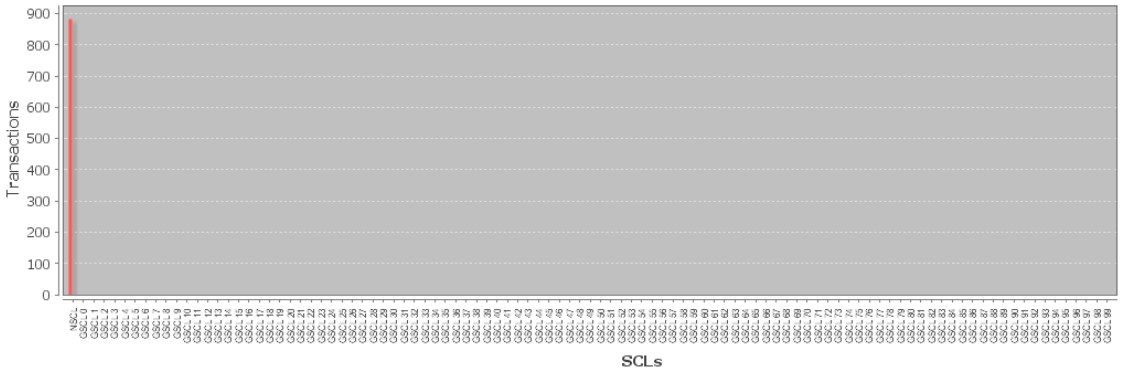

Figure 4.8: Transactions distribution without OSCL

The second simulation is performed in an OSCL-enabled system however the caching is disabled. Fig. 4.9 shows that the number of NSCL transactions decreased to 165 compared to the previous simulation and the average number of transactions per GSCL is about 20 with a maximum of 100 transactions in GSCL_11 and a minimum of 5 transactions in GSCL_98. It is clear here, that the OSCL enables one to balance the load between all existing nodes instead of concentrating it on the

NSCL.

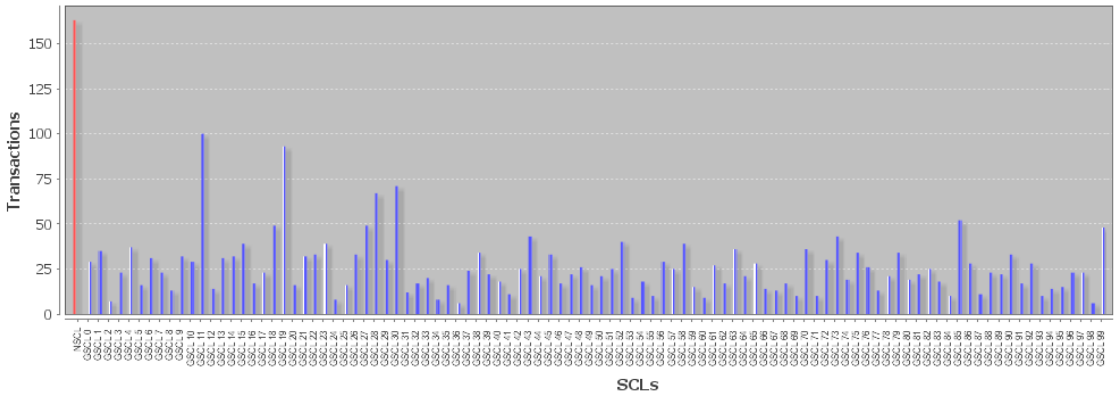

Figure 4.9: Transactions distribution using OSCL

The third simulation is performed on an OSCL-enabled system with caching. Fig. 4.10 shows that the number of NSCL transactions decreased to 135 and the average number of transactions per GSCL decreased 15 compared to the previous simulation with a maximum of 48 transactions in GSCL_27 and a minimum of 2 transactions in GSCL_14. It is clear here, that a OSCL with caching enables a better dissemination of data and networking so better system scale.

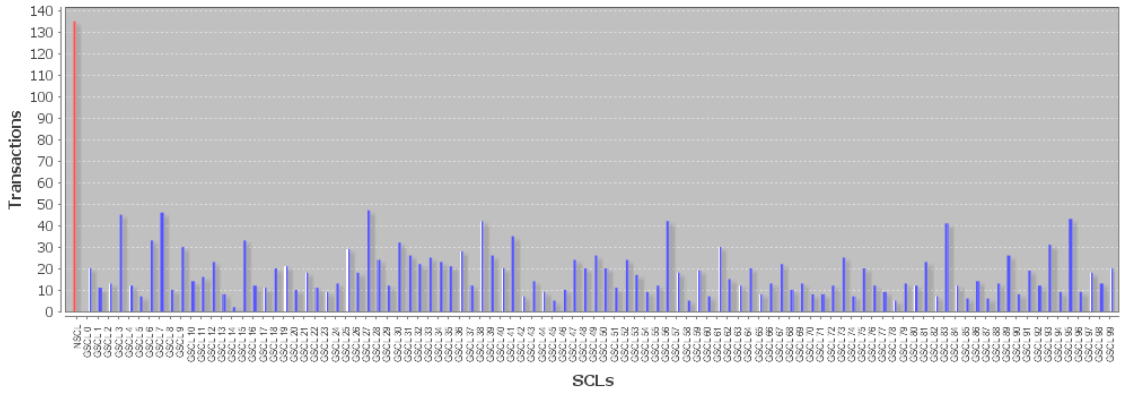

Figure 4.10: Transactions distribution using OSCL with caching

4.6 Dynamic random graph model for diameter constrained networks

OSCL enables one to build advanced networked systems, made of interacting dynamic entities. However the properties of such interacting system cannot be deterministically known in advance making very hard sizing the network resources. In this study we propose a graph-based theoretical model to assess, predict, and control the performance of the OSCL network. The proposed model enables one to compute relevant network parameters such as the average graph degree, network robustness, transient time, and link failures.

4.6.1 Target Scenario and Notation

The target scenario considered in this section consists of a Dynamic random graph (DRG) of N vertices, n_q being the q-th vertix ($q \in [1, N]$). Furthermore, an ordered sequence of equi-probable couples of vertices is considered, among which a path composed of no more than D edges has to be established. The t-th couple in the ordered sequence of vertices is described by (n_{i_t}, n_{j_t}). For the sake of simplicity, the variable t will be referred to as *time* from now on. Knowing the t-th couple, a new edge is established in the graph if and only if the two vertice (n_{i_t}, n_{j_t}) are not reciprocally reachable in no more than D edges.

It is worth to note that equi-probable arrivals (i.e., homogeneous conditions) are usually assumed in the current literature dealing with diameter constrained graphs [122]. In fact, if the vertice of the graph represent the gateway through which the service of a large number of nodes are made available (as in M2M systems [102]), it is not unlikely that the overlay that inter-connect such gateways actually reflects this assumption.

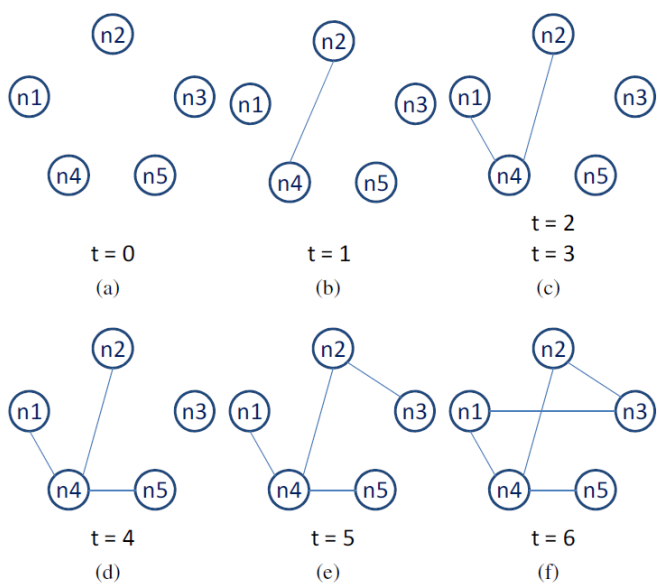

Figure 4.11: Evolution of a random graph ($N = 5$, $D = 2$)

To provide an illustrative example of the networked system we are modeling as a random graph, Fig. 4.11 shows the evolution of a graph made of $N = 5$ vertice and constrained by $D = 2$ max path length. In this example, the ordered sequence of vertice (n_{i_t}, n_{j_t}) that ask for a path is: (n_2, n_4), (n_1, n_4), (n_1, n_2), (n_4, n_5), (n_2, n_3), and (n_1, n_3). Accordingly, for each of them, a new edge is added if and only if a path shorter than 3 edges is not already available among corresponding vertice. In the sequel of the contribution, we will derive a law that rule the evolution of this kind of graphs in a general case.

Figure 4.11 shows an example of evolution of a random graph ($N = 5$, $D = 2$) as follows:

(a) initial state.

(b) the first edge is added at $t = 1$ to connect vertice n_2 and n_4.

(c) the second edge is added at $t = 2$ to connect vertice n_1 and n_4, but no edge is added at $t = 3$ because the vertice n_1 and n_2 (asking for a connection) are already connected by a path of 2 hops $\leq D$.

(d) the third edge is added at $t = 4$ to connect vertice n_4 and n_5.

(e) the fourth edge is added at $t = 5$ to connect vertice n_2 and n_3.

(f) the fifth edge is added at $t = 6$ to connect vertice n_1 and n_3, for which the only existing path was longer than D hops.

Table 4.1: Notation.

Symbol	**Meaning**
N	Number of vertice
k_t	Average degree at time t
n_q	q-th vertex
D	Maximum diameter
(n_{i_t}, n_{j_t})	t-th couple of vertices wishing to establish a path
l_t	Number of edges at time t
w_t	Number of path with less than $D + 1$ edges between a couple of vertices at time t
P_t	Probability to find a path no longer than D edges between the couple (n_{i_t}, n_{j_t}) at time t
λ	Average number of arrivals per time t
A_t^{NxN}	Symmetric binary adjacency matrix at time t
$Pr\{x\}$	Probability of event x
$\hat{x}$	Upper bound on x

4.6.2 Dynamics of The Average Graph Degree

Proposition *For a sufficiently large N, the following expression describes the dynamics of the average graph degree:*

$$k_{t+1} \approx k_t + \frac{2}{N} \cdot exp\left(-\frac{1}{N} \cdot \frac{k_t^{D+1} - k_t}{k_t - 1}\right) \tag{4.1}$$

Proof *P_t expresses the probability to find a path at time t (no longer than D edges) between a generic couple of vertices $(n_{i_{t+1}}, n_{j_{t+1}})$, knowing that the number of already existing edges is l_t. Since we are assuming homogeneous conditions, P_t is the same for all the possible couples (n_{i_t}, n_{j_t}). The average number of arrivals per time t is λ. Accordingly, our model is grounded on*

$$l_{t+1} = l_t + \lambda \cdot P_t \tag{4.2}$$

Given that the rate of arrival λ is constant equal to 1, we can write

$$l_{t+1} = l_t + P_t \tag{4.3}$$

which, considering that the average degree [123] (i.e., the number of edges per vertex) is $k_t = \frac{2 \cdot l_t}{N}$, can be also expressed as:

$$k_{t+1} = k_t + \frac{2}{N} \cdot P_t \tag{4.4}$$

The next step is to find an accurate approximation for the probability P_t starting from the binary symmetric adjacency matrix A_t^{NxN}. In this latter, $A_t(i,j) = A_t(j,i) = 1$ if and only if an edge between n_i and n_j exists at time t, otherwise $A_t(i,j) = A_t(j,i) = 0$. The presence of an edge between any couple of vertices at time t will be expressed using the A_t^{NxN} matrix.

To fulfill this objective, we first leverage a well known property of the matrix A_t: $A_t^c(i,j) = 0, c \in \mathbb{N}^+$, if and only if no path composed of c edges exist between n_i and n_j at time t [124]. In this way, without lack of generality, P_t can be expressed as follows:

$$P_t = \prod_{c=1}^{D} Pr\{A_t^c(i,j) = 0\} \tag{4.5}$$

Since any element of A_t^c is no other than the sum of N^{c-1} elements, each one being a product of c coefficients $A_t(i,j)$ belonging to A_t, we can approximately write:

$$Pr\{A_t^c(i,j) = 0\} \approx Pr\{(\sum_{1}^{N^{c-1}} \prod_{1}^{c} A_t(i,j)) = 0\} \tag{4.6}$$

The probability of the event "the sum of N^{c-1} positive elements is equal to 0" is equivalent to the intersection of the probabilities of the events "each element of N^{c-1} is equal to 0". Since we are considering equiprobable and independent element, we can write:

$$Pr\{A_t^c(i,j) = 0\} \approx [Pr\{(\prod_{1}^{c} A_t(i,j)) = 0\}]^{N^{c-1}} \tag{4.7}$$

Recalling that for any event E we have $Pr(E) = 1 - Pr(\overline{E})$, and since $A_t(i,j)$ belongs to [0,1], we can write:

$$Pr\{A_t^c(i,j) = 0\} \approx [1 - Pr\{(\prod_{1}^{c} A_t(i,j)) = 1\}]^{N^{c-1}} \tag{4.8}$$

The probability of the event "the product of c elements is equal to 1" is equivalent to the intersection of the probabilities of the events "each element of c is equal to 1". Since we are considering equiprobable and independent element, we can write:

$$Pr\{A_t^c(i,j) = 0\} \approx [1 - (Pr\{A_t(i,j) = 1\})^c]^{N^{c-1}} \tag{4.9}$$

Given that the graph is undirected, there are $2 \cdot l_t$ elements that are equal to 1 in A_t, it yields $Pr\{A_t(i,j) = 1\} = \frac{2 \cdot l_t}{N^2}$. We can write:

$$Pr\{A_t^c(i,j) = 0\} \approx \left[1 - \left(\frac{2 \cdot l_t}{N^2}\right)^c\right]^{N^{c-1}} \tag{4.10}$$

Recalling that $(1 + \frac{x}{n})^n \to e^x$, when $n \to \infty$, thus for a sufficiently large N we obtain the following new approximation:

$$Pr\{A_t^c(i,j) = 0\} \approx exp\left(-N^{c-1} \cdot \left(\frac{2 \cdot l_t}{N^2}\right)^c\right) \tag{4.11}$$

Accordingly, by substituting (4.11) in (4.5), we obtain:

$$P_t \approx exp\left(-\sum_{c=1}^{D} N^{c-1} \cdot \left(\frac{2 \cdot l_t}{N^2}\right)^c\right) \tag{4.12}$$

which is equivalent to $P_t \approx exp\left(-1/N \cdot \sum_{c=1}^{D} \left(\frac{2 \cdot l_t}{N}\right)^c\right)$. This latter expression is no other than a geometric series with parameter $\left(\frac{2 \cdot l_t}{N}\right)$, multiplied by $-1/N$, truncated at the first D terms, and without the first addend 1, which, after a little algebra, can be expressed as:

$$P_t \approx exp\left(-\frac{1}{N} \cdot \frac{(\frac{2 \cdot l_t}{N})^{D+1} - \frac{2 \cdot l_t}{N}}{\frac{2 \cdot l_t}{N} - 1}\right) \tag{4.13}$$

From [123], the average degree can be expressed as $k_t = \frac{2 \cdot l_t}{N}$. So that we obtain the proof by substituting (4.13) in (4.4).

$$k_{t+1} \approx k_t + \frac{2}{N} \cdot exp\left(-\frac{1}{N} \cdot \frac{k_t^{D+1} - k_t}{k_t - 1}\right) \tag{4.14}$$

This end the proof.

4.7 DRG model simulations

To validate the model, we have considered a complex scenario composed of N vertice (with N up to 2^{16}) and D ranging from 3 to 10. Using an ad hoc simulator we developed in Matlab, the relative error between the real evolution of the degree $k(t)$ and the one estimated using (4.14) for all t is evaluated. In any case, we found that the average relative error is below 10% if we consider the entire evolution of $k(t)$. Also, the relative error at steady state (once the graph is completely formed), for $D \leq 5$, is below 10%, whatever N. Finally, we notice a slight increase in the relative error as D increases: (in any case) it remains smaller than 25% and it falls below 20% for $N > 2^{12}$ (see Fig. 4.12 and 4.13). These results are expected because the model is based on two main approximations, used to derive (4.6) and (4.11), respectively. The accuracy of the first (resp. second) approximation decreases by increasing (resp. decreasing) D (resp. N). As a consequence, it is quite straightforward to observe that estimation errors increase by increasing D whereas they decrease by increasing N.

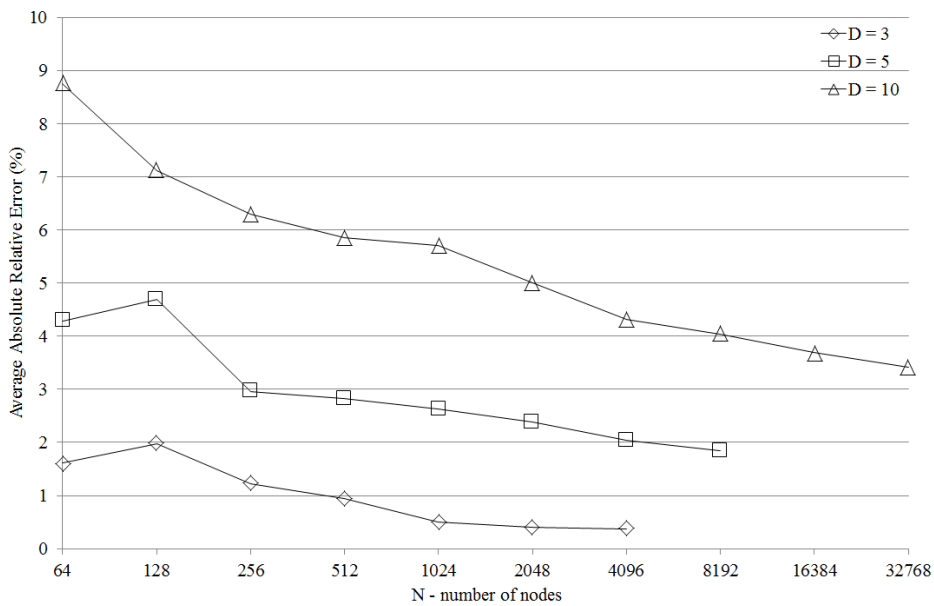

Figure 4.12: Average absolute relative errors of the model (4.14).

To provide an illustrative example, Fig. 4.14 plots the dynamic evolution of the number of edges versus the time t (similar results have been obtained for different values of N and D so that the average relative error is below 10% in all cases).

We remark that knowing k in advance allows one to size the processing and communication capabilities of the physical nodes of the overlay. In fact, the higher k is the higher the load due to packet processing and relaying. So that, an error by x% on the estimated number of links per node means that physical nodes should be over provisioned x% more with respect to the outcomes of the model.

It is worth to note that the number of edges linearly increases with t till a saturation point is

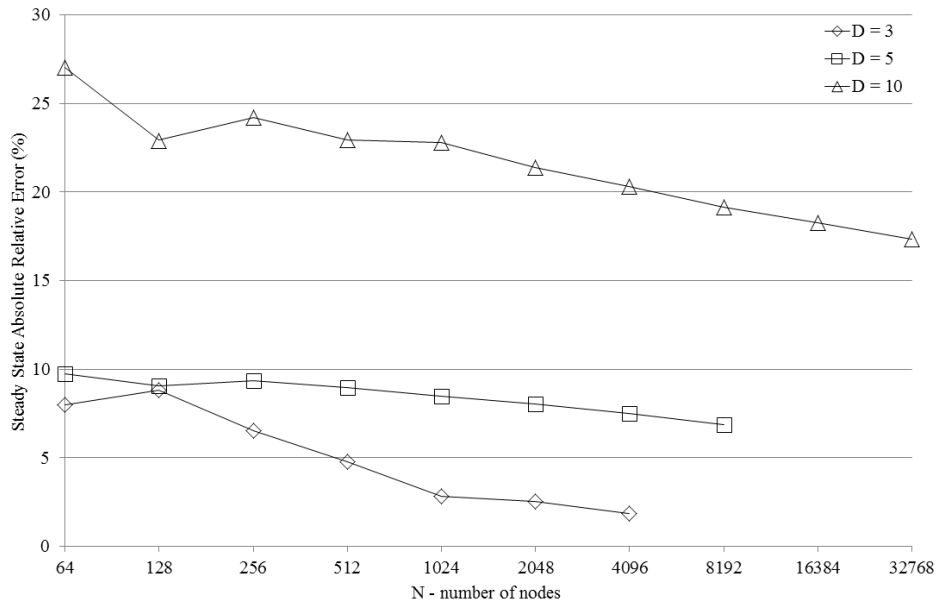

Figure 4.13: Steady State absolute relative errors of the model (4.14).

reached. From that moment on, l exhibits a very slow rise. This can be explained by plotting also the values of the probability P_t. Fig. 4.15 shows that P_t is almost equal to one for some time during the network formation, meaning that, since the number of edges is low, it is highly likely to add a new edge as soon as a new couple of nodes needs to establish a path. At the same time, the values of P_t abruptly decreases after a certain t, meaning that the topology reached the steady state.

4.8 DRG model example applications

In this section we show different example application for measuring OSCL properties such as rank bound, robustness, transient duration and link failures.

4.8.1 Rank bound and comparison

Theorem 1 *Being $\hat{k}$ the maximum degree k in system (4.14), it can be bounded as follows:*

$$\hat{k} \leq \sqrt[D]{2 \cdot N \cdot lnN}. \tag{4.15}$$

Proof *The way we are building the overlay is so that an edge between a couple of vertices (n_{i_t}, n_{j_t}) is formed or not based only upon the first time that couple issues a request for a path. If a path shorter than $D+1$ hops already exists the edge is not established otherwise it is established. From that moment on, the next requests for a path issued by the same couple of vertices will not sort any effect.*

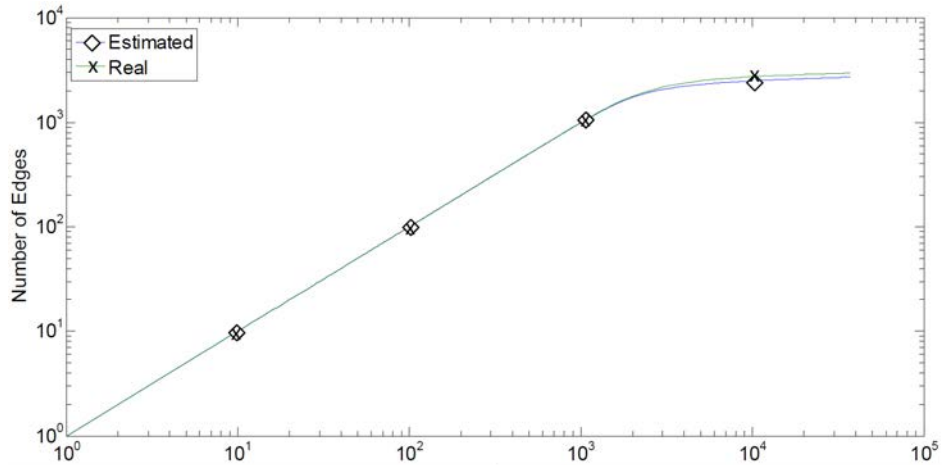

Figure 4.14: Evolution of the number of edges over the time t, ($N = 1000$, $D = 5$).

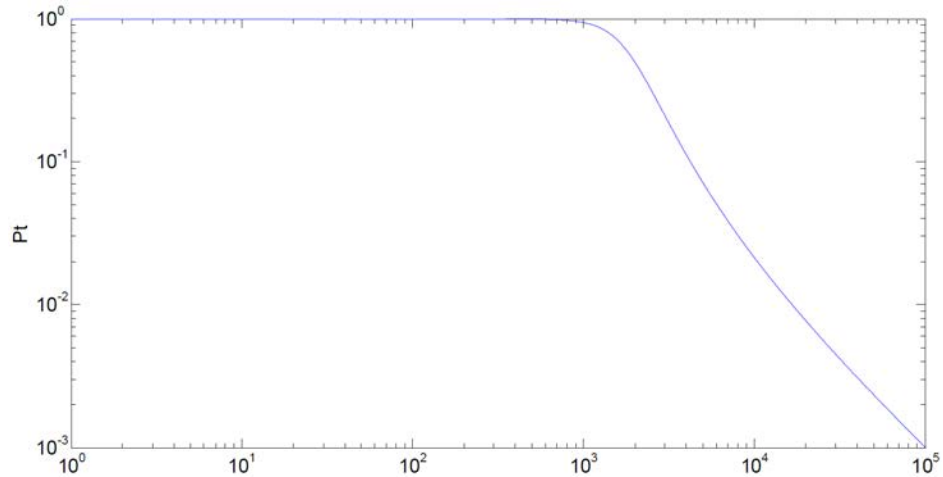

Figure 4.15: Evolution of P_t, ($N = 1000$, $D = 5$).

Based on this consideration, we extract from the sequence of equi-probable couples of vertice considered in the study, the sequence of instants in which any couple of vertices appears for the first time. Of course, the length of such a sequence of time instants will be composed of at most $N(N-1)/2$ *elements.*

In order to estimate an upper bound on the steady state average degree $\hat{k}$*, it is necessary to consider that at time* t*,* $1/P_t$ *expresses the average time required to establish the next edge in the overlay.*

Under this assumption, the expression of P_t *to consider is slightly different from that in Eq. (4.13) because the sequence of vertix couples we are considering to proof this Theorem is chosen in such a way that no one edge path exists at time* t *for* (n_{i_t}, n_{j_t})*. The avoid ambiguity, we will refer to* P'_t *to refer to this new probability. Accordingly, it follows:*

$$P'_t = \prod_{c=2}^{D} Pr\{A_t^c(i_{t+1}, j_{t+1}) = 0\} \tag{4.16}$$

which, following the same passages reported in the proof of Proposition 1, can be also written as:

$$P'_t \approx exp\left(-\frac{1}{N} \cdot \frac{(\frac{2 \cdot l_t}{N})^{D+1} - (\frac{2 \cdot l_t}{N})^2}{\frac{2 \cdot l_t}{N} - 1}\right) \tag{4.17}$$

Therefore to estimate an approximated upper bound on $\hat{k}$*, it is sufficient to find any value of* k_t *so that* $\frac{1}{P'_t} \geq N(N-1)/2$*, which gives* $\sum_{D}^{c=2} k_t^c \geq 2 \cdot N \cdot lnN - N \cdot ln2$*. Notice that the latter inequality is satisfied if* $k_t^D \geq 2 \cdot N \cdot lnN$*, from which the proof follows. This end the proof.*

Remark 1 *It is worth remarking that, the bound on* $\hat{k}$ *can be fruitfully exploited to compare the properties of the graph under study with respect to well known graph. To this end, if we consider as ground for comparison the delay optimal de Bruijn graphs, representing one of the most compact topologies discovered so far [122], we will find that, for the same diameter* D*, the degree of the unstructured overlay considered in this work is only* $\sqrt[D]{2 \cdot lnN}$ *times larger at most, even if it is based on a much simpler construction mechanism.*

4.8.2 Robustness

Knowing that the graph construction model adopted herein ensures, at steady state, at least one path shorter than $D+1$ edges between any couple of nodes, it is worth investigating how many paths (composed of less than $D+1$ edges) are present between any couple of nodes. This metric is intimately related to the topology robustness: the higher the number of paths the higher the number of alternative solutions to route messages in case of failures.

Theorem 2 *Defined as* w_t *the number of paths composed of less than* $D+1$ *edges between any couple of nodes at time* t*, it holds:*

$$w_t = \frac{1}{N} \cdot \frac{k_t^{D+1} - k_t}{k_t - 1} \tag{4.18}$$

Proof *Also in this case the properties of the adjacency matrix are exploited. In particular, $A_t^c(i,j)$ indicates the number of paths composed of c edges between the vertice n_i and n_j. We have:*

$$w_t = \sum_{s=1}^{D} Pr\{A_t^s(i,j) = 1\} \tag{4.19}$$

which can be written as:

$$w_t = \sum_{s=1}^{D} (Pr\{A_t(i,j) = 1\})^s \tag{4.20}$$

recalling that $Pr\{A_t(i,j) = 1\} = \frac{2 \cdot lt}{N^2}$, we obtain:

$$w_t = \sum_{s=1}^{D} (\frac{2 \cdot lt}{N^2})^s \tag{4.21}$$

recalling that $k_t = \frac{2 \cdot lt}{N}$, we obtain:

$$w_t = \frac{1}{N} \sum_{s=1}^{D} k_t^s \tag{4.22}$$

with a little algebra we obtain:

$$w_t = \frac{1}{N} \cdot \frac{k_t^{D+1} - k_t}{k_t - 1} \tag{4.23}$$

This end the proof.

Based of this Theorem, we can derive an approximate assessment of the level of redundancy, if we consider for k_t, the bound derived in Theorem (1), $k_t = \sqrt[D]{2 \cdot N \cdot lnN}$. Under this assumption, we can obtain:

$$\hat{w} = \frac{1}{N} \cdot \frac{\sqrt[D]{2 \cdot N \cdot lnN}}{\sqrt[D]{2 \cdot N \cdot lnN} - 1} \cdot (2 \cdot N \cdot lnN - 1) \tag{4.24}$$

For $N > e$, we have $\sqrt[D]{2 \cdot N \cdot lnN} > 1$, thus $\frac{\sqrt[D]{2 \cdot N \cdot lnN}}{\sqrt[D]{2 \cdot N \cdot lnN} - 1} > 1$, and so we can write:

$$\hat{w} > 2 \cdot lnN - 1/N \tag{4.25}$$

Thus, for $N > e^2$, $\hat{w} = 2 * ln(e^2) - 1/(e^2) = 3.86$. Based of this result, we can say that the overlay investigated in this work is able to provide at steady state at least three paths among any couple of vertices.

4.8.3 Transient duration

From (4.14), l_t monotonically increases with t. At the same time k_t is upper bounded, according to Theorem 1. As a consequence, we can conclude that the system (4.14) is convergent. Herein, the transient behavior of k_t is investigated to provide a more complete description of the network overlay dynamics.

Proposition *If we define the transient duration t_t of the system (4.14) as the time instant for which $P_{t_t} \geq \epsilon$, for any given $\epsilon > 0$, with ϵ small enough, it yields:*

$$t_t \leq \frac{N}{2} \sqrt[D]{N \cdot ln(1/\epsilon)} \tag{4.26}$$

Proof *During the transient of the system (4.14), $l_t \approx t$ because the number of already existing edges is so low that any new couple of vertices that wish to establish a path will trigger the creation of a new link (see also Fig. 4.14 for an example). To define the transient duration t_t, we consider that $P_{t_\alpha} \geq \epsilon$ for any given $t_\alpha \in [0, t_t]$ and any $\epsilon > 0$, with ϵ smaller enough.*

$P_{t_\alpha} \geq \epsilon$ can be translated to :

$$exp\left(-\frac{1}{N}\frac{k_{t_\alpha}^{D+1} - k_{t_\alpha}}{k_{t_\alpha} - 1}\right) \geq \epsilon \tag{4.27}$$

which can be written as:

$$\frac{k_{t_\alpha}^{D+1} - k_{t_\alpha}}{k_{t_\alpha} - 1} \leq -N \cdot ln(\epsilon) \tag{4.28}$$

with a little algebra we obtain:

$$\sum_{s=1}^{D} k_{t_\alpha}^{s} \leq N \cdot ln(1/\epsilon) \tag{4.29}$$

which can be also written as:

$$k_{t_\alpha}^{D} \leq N \cdot ln(1/\epsilon) \tag{4.30}$$

then we obtain:

$$k_{t_\alpha} \leq \sqrt[D]{N \cdot ln(1/\epsilon)} \tag{4.31}$$

recalling that $k_t = \frac{2 \cdot lt}{N}$, we obtain:

$$l_{t_\alpha} \leq \frac{N}{2} \sqrt[D]{N \cdot ln(1/\epsilon)} \tag{4.32}$$

recalling that, during the transient, $l_t \approx t$, we can write:

$$t_\alpha \leq \frac{N}{2} \sqrt[D]{N \cdot ln(1/\epsilon)} \tag{4.33}$$

finally, given that $t_\alpha \in [0, t_t]$, we obtain:

$$t_t \leq \frac{N}{2} \sqrt[D]{N \cdot ln(1/\epsilon)} \tag{4.34}$$

This end the proof.

4.8.4 Link failures

To include also possible link failures and dynamics in the model, it is necessary to modify Eq. (4.2) as follows:

$$l_{t+1} = l_t + P_t - \lambda_o \cdot l_t \tag{4.35}$$

where λ_o is the probability that an edge is removed during one time step. The resulting equations could be very useful to design topology management algorithms using control theoretic arguments. Its utility in finding the uniquness of the equilibrium point is shown in the following Theorem.

Theorem 3 *The system (4.35) admits one and only one equilibrium point $l = l_\infty$.*

Proof *In order to find the equilibrium point of system (4.35), we impose $l_{t+1} = l_t = l_\infty$ in (4.35). Accordingly, the following equality is obtained:*

$$P_\infty = \lambda_o \cdot l_\infty \tag{4.36}$$

Eq. (4.36) admits only one solution because its leftmost member monotonically decreases with l, starting from the value one at $l = 0$ whereas the rightmost member monotonically increases, starting from zero at $l = 0$.

This ends the proof.

4.9 Conclusion

This chapter presented a state of the art about network overlays and summarized the main features of the ICN and NDN architectures. The OSCL solution, which offers a meshed M2M network and makes use of the NDN architecture for an efficient networking data distributed is described and significant use cases are illustrated. An NDN simulator for OSCL called OSCLsim is designed and developed to validate the proposed approach. A DRG model for diameter constrained network enabling to assess, predict, and control the performance of the OSCL network is presented, simulated, and validated and useful examples of its applications are described.

Chapter 5

Conclusion and perspectives

In this chapter we provide an overview of the main contributions and results carried out and developed during this thesis. We present also a set of perspectives of enhancements and extensions, and gives an outlook into new research areas to follow up this work.

Conclusion

This study addressed the need for a horizontal M2M service platform to overcome the current M2M market fragmentation. Only a standard-based solution with a modular and extensible operational architecture can solve such a big challenge. In addition, interoperability must be achieved in terms of communications and semantics and the platform must be flexible enough to work in constrained environment. Addressing this challenge, we designed, implemented, and experimented the OM2M platform offering a flexible and extensible system architecture for M2M interoperability based on the SmartM2M standard. We also defined and experimented a new resource structure model offering small payload to meet the requirement of constrained devices. In addition, we designed the IoT-O ontology model by merging together existing ontologies to achieve semantic interoperability between heterogeneous applications and devices.

OM2M provides a promising solution to realize innovative and complex scenarios through cross-domain interoperability, however keeping such heterogeneous and dynamic system alive is costly in terms of time and money. Without autonomic computing capabilities, any M2M system will be quickly out of control. To address this challenge, we designed, implemented, and integrated the FRAMESELF framework to retrofit self-management capabilities in M2M system. Inspired by the Autonomic Computing paradigm, FRAMESELF proposes a complete MAPE-K architecture design to dynamically adapt the OM2M system behavior according to the environment changes. In addition, semantic rules are defined to enable dynamic device and applications discovery, semantic matching, and resource architecture reconfiguration based on the IoT-O ontology.

Hiding intrinsic complexity of M2M systems paves the way to mass-scale deployment of M2M devices in various domains. However the current internet infrastructure was never designed for this kind of usages. New networking protocols and data dissemination mechanisms should be conceived to follow such a phenomenal growth. To address this challenge, we designed, simulated and validated the OSCL approach, a decentralized organization structure relying on a meshed network topology as an alternative to the traditional centralized approach. OSCL is based on the NDN architecture

and supports multi-hop communication and distributed caching. We designed and implemented also OSCLsim, an NDN simulator to validate the proposed approach. In addition, we formulated a theoretical model based on random graph theory to describe the evolution of the distributed M2M network average degree, network robustness, transient time, and link failure.

First, OM2M was experimented in the ITEA2-A2NETS [1] European project as a horizontal M2M service platform to interconnect heterogeneous devices and applications pertaining to three different business cases: Smart Metering, Well being, and car sharing. As a proof of concept we built an interworking demonstrator including (1) a smart building mockup equipped with Zigbee devices, (2) a connected car equipped with a controller area network (CAN) bus, and (3) an electric bicycle equipped with bluetooth low energy (BLE) sensors. Using OM2M, the three domains are merged together setting a path towards new services and possibilities for users.

Second, OM2M was experimented in the ADREAM [2] smart building and living lab of LAAS-CNRS. ADREAM is a complex cyber-physical system composed of a myriad of vendors-specific devices including heating, ventilating, and air conditioning (HVAC) devices, heat pump system, photo-voltaic cells, weather station, robots, cameras, luminaries, smart meters and many other specific sensors and actuators. ADREAM resources were conceived to be shared by many research teams with different specializations. So OM2M was used as an abstraction layer to hide the complexity of this building. Semantic capabilities of OM2M enable researchers to access the building devices and share their own data in a seamless way.

Now, OM2M is being experimented and used by different industrial companies around the world to realize both internal interworking for their own environment, but also to build horizontal M2M systems for more advanced cross-domain interoperability. OM2M features and capabilities are about to be extended through research activities and thesis projects in various French laboratory as well as abroad. The OM2M architecture is currently serving as teaching materials for IoT courses in the National Institute of Applied Sciences (INSA) in Toulouse and the National Chiao Tung University (NCTU) in Taipei.

Since October, 2013, the OM2M platform was maintained and distributed within the Eclipse Foundation as the first open source implementation of the SmartM2M standard [3]. The OM2M first release [4] was published in April, 2015. Results and products established in this PhD were published in 7 international journals, 8 international conferences, 3 scientific magazines, and 13 project deliverables. They paved the way to significant contributions to SmartM2M and OneM2M standards.

Perspectives

Research conducted during this thesis helped addressing big challenges for M2M such as interoperability, complexity and scalability, and have given birth to a set of software products, tools and models. Given, that the IoT is a constantly and rapidly evolving field, several extensions and improvements can be done as future work to current results. Some perspectives are listed below.

[1] Autonomic Services in M2M networks project website: https://itea3.org/project/a2nets.html

[2] Dynamic reconfigurable architectures for embedded mobile autonomous systems (ADREAM) website: https://www.laas.fr/public/en/adream

[3] Eclipse OM2M project proposal: https://www.eclipse.org/proposals/technology.om2m

[4] Eclipse OM2M press release: https://www.eclipse.org/org/press-release/20150408_om2m.php

OM2M migration to OneM2M standard

As future work, we propose to migrate the OM2M platform to the OneM2M standard, which was published in January 31, 2015. Given that SmartM2M deeply influenced the OneM2M standard, we estimate the amount of work required to do the migration to 30%. Main improvements defined in OneM2M are (1) the specification of a new type of node called Middle Node (MN), which enables one to create more advanced hierarchical M2M networks, (2) Definition of non blocking requests, which can be both synchronous or asynchronous to add more flexibility in the way of exchanging data, (3) definition of new reference point called "mcn" to link the service layer with the underlying network. We planned to release the second version of OM2M compliant with the OneM2M standard in September 30, 2015.

OM2M capabilities extension

The MQTT protocol will be integrated as a new communication binding to provide lightweight asynchronous communication based on the publish/subscribe pattern. It comes with three QoS levels for quality of service options and enables bandwidth efficiency to lower network costs and reduce traffic loads. The LWM2M protocol will be integrated to enable device management and service enablement over constrained sensor networks. In addition, it offers a light and compact secure communication interface along with an efficient data model that meets the requirements of vertical application. We envisage also to reinforce OM2M security by implementing the TLS-PSK protocol, which relies on symmetric keys shared in advance among the communicating parties. It offers a set of cryptographic protocols dedicated for low power and lossy networks.

OM2M deployment in smart cities

Recently, LAAS-CNRS participated, through OM2M, in many IoT research project calls covering a variety of application domain, which will enable to extend and experiment more the capacity of our platform. In particular, it participated with cities of Toulouse and Hamburg to a project call for Smart Cities with the aim of real deployment of OM2M in different European cities.The main goal here is to enhance performance and wellbeing, to reduce costs and resource consumption, and to engage more effectively and actively with citizens. Sectors include energy, transport, health care, water and waste.

OneM2M architecture improvement

The OneM2M standard already supports non hierarchical resource structure, however it lacks of the collection resource, which is an essential pattern to enable HATEOAS (Hyper-media As the Engine of Application State) maturity level for REST architecture. Using the current OneM2M architecture, a user will not be able to find his way on the API simply by reading hyperemia and following links. A human-readable documentation is required instead, which impedes the maintenance and future evolution of the platform. In this sense, a set of contributions are already submitted and discussed with the OneM2M technical committee. Additional technical contributions will be submitted to apply the necessary changes.

OneM2M semantic integration

Contrary to the SmartM2M standard, OneM2M provides a native support for semantics, however there has not been any significant progress on this issue for the first specification release. For this reason, we will try to contribute our IoT-O ontology concepts and relationships into the OneM2M core ontology. Two options are available. The first one is reference-based integration. It uses the semantic attribute "ontologyRef", already defined in the standard, to point the application to a remote location where it can find the full definition of the ontology. It can also be used to serve as just a reference to an ontology supposedly known by the application (i.e. through the use of predetermined ontology catalog). The second option is inline integration, consists of using the Resource Description Framework (RDF) as syntax notation and data serialization format to expose and share across different applications. Using this option, any application can understand the received content since it already includes concepts and relationships. Such format is not yet defined in oneM2M standard, but can be pushed in future oneM2M release.

FRAMESELF improvement and extension

FRAMESELF provides an architectural approach for designing autonomic managers, which are the basic building blocks of an autonomic system. Autonomic managers can be considered as services within service oriented architecture and may share different type of relationships between each others such as cooperation or competitions. Lacking global control, an autonomic manager may oscillate resulting in an unstable state. Inspired form the multi-agent systems, a set of communication protocols could be defined to enable efficient organization, synchronization, and negotiations between autonomic managers. In addition, FRAMESELF is designed to be generic. So it is easy to introduce some advanced algorithm and reasoning capabilities relying on artificial intelligence algorithms, machine learning, and constrained satisfaction problems.

Towards meshed OneM2M architecture

Even with the integration of middle nodes, the OneM2M architecture remains centralized. In other terms, it is not possible to create a meshed topology using OneM2M, which is not efficient for the overall system scalability. As a solution, we propose to move to a more decentralized network by authorizing the creation of meshed overlay of nodes and making use of NDN naming and routing techniques. We will also extend, the OSCLsim simulator in order to support the new OneM2M resource structure, networking and system architecture. Finally, we will enhance our DRG model to deal with caching and heterogeneous conditions.

List of publications

International journals

- Ben Alaya M, Medjiah S, Monteil T, Drira K, Towards Semantic Data Interoperability in oneM2M Standard. IEEE Communications Magazine, IEEE, 9p, 2015
- Grieco LA, Ben Alaya M, Monteil T, Drira K, Diameter Constrained Overlays with Faulty Links: Equilibrium, Stability, and Upper Bounds. IEEE Transactions on Circuits and Systems TCAS II, 8p, 2015.
- Grieco LA, Ben Alaya M, Monteil T, Drira K, A Dynamic Random Graph Model for Diameter-Constrained Topologies in Networked Systems. IEEE Transactions on Circuits and Systems TCAS II, vol. 61, pp. 982-986, 2014.
- Latvakoski J, Ben Alaya M, Ganem H, Jubeh B, Iivari A, Leguay J, Bosch J, Granqvist N,Towards Horizontal Architecture for Autonomic M2M Service Networks. MDPI Future Internet, vol. 3, no. 4, pp. 130-173, 2014.
- Latvakoski J, Iivari A, Vitic P, Jubeh B, Ben Alaya M, Monteil T, and al, A Survey on Autonomic M2M Service Networks. MDPI Computers, vol. 3, no. 4, pp. 130-173, 2014.
- Iivari A, Vaisanen T, Ben Alaya M, Riipinen T, Monteil T, Harnessing XMPP for Machine-to-Machine Communications & Pervasive Applications. Journal of Communications Software and Systems, vol. 10, no. 3, pp. 163-178, 2014.
- Ben Alaya M, Monteil T, FRAMESELF: an ontology-based framework for the self-management of machine-to-machine systems. WILEY Concurrency and Computation: Practice and Experience, Vol 25, September, vol. 27, no. 6, pp. 1412-1426, 2015.
- Gharbi, G. Ben Alaya, M. Diop, C. Exposito, E, AODA: An automatic and ontology-driven architecture for service-oriented and event-driven systems. International journal of collaborative enterprise, vol. 3, pp. 167-188, 2013.

International conferences

- Vogli E, Ben Alaya M, Monteil T, Grieco LA, Drira K, An efficient resource naming for enabling constrained devices in SmartM2M architecture. IEEE ICIT International Conference on Industrial Technology, 8p. 2015.

- Ben Alaya M, Banouar Y, Monteil T, Chassot C, Drira K. OM2M: Extensible ETSI-compliant M2M service platform with self-configuration capability. RAMCOM, Recent Advances on M2M comm. Hasselt, Belgium, vol. 32, no. 0, pp. 1079-1086, June 2014.

- Grieco LA, Ben Alaya M, Monteil T, Drira K. Architecting Information Centric ETSI-M2M systems. IEEE International Conference on Pervasive Computing and Communications WIP, pp. 24-28, Budapest Hungary, pp. 211-214, March 2014.

- Ben Alaya M, Matoussi S, Monteil T, Drira K. Autonomic computing system for self-management of machine-to-machine networks. ACM Self-IoT, international workshop on Self-aware internet of things. San Jose, USA, pp. 25-30, June 2013.

- Ben Alaya M, Monteil T. FRAMESELF: A Generic Context-Aware Autonomic Framework for Self-Management of Distributed Systems, IEEE WETICE, Enabling Technologies: Infrastructure for Collaborative Enterprises. pp. 60-65, June 2012.

- Ben Alaya M, Monteil T, Drira K. Autonomic framework based on semantic models for self-management of ubiquitous systems. ACM UbiComp 2012, Ubiquitous Computing. Pittsburgh, USA. pp.860-862. September 2012.

- Gharbi G, Ben Alaya M, Diop C, Exposito E. An Autonomic and Ontology-Driven Architecture for Service-Oriented and Event-Driven Systems. IEEE WETICE, Enabling Technologies: Infrastructure for Collaborative Enterprises. pp. 167-188. June 2012.

- Ben Alaya M, Baudin V, Drira K. Dynamic Deployment of Collaborative Components in Service-Oriented Architectures. IEEE NOTERE, New Technologies of Distributed Systems. Paris, France. pp. 1-8. May 2011.

Magazines and press releases

- "Le Monde, Science & Techno", "Robots and Humans, roommates in Toulouse", pp. 4-5, January 06, 2012.

- "Le Parisien", Cameras and sensors everywhere", pp. 1-3, July 18, 2012.

- "La Dépêche du midi", Energy smart management via Machine-to-Machine communications, pp. 21, October 14, 2012.

- Eclipse OM2M press release: "Eclipse Announces First Release of Eclipse OM2M Project", April 08, 2015. URL: https://www.eclipse.org/org/press-release/20150408_om2m.php

Webinars

- Eclipse OM2M webinar: "OM2M: Standardized service platform for M2M interoperability", October 14, 2014. URL: https://www.youtube.com/watch?v=WKwPfbdMUHw

Standards specifications

- Editorship of ETSI TS 103315 V1.1.1 (2015-01); SmartM2M; Interoperability Test Specification for ETSI M2M Primitives.
- Contribution to SmartM2M TS 102 921: Correct re-targeting mechanism for the HTTP binding. 2015 (Ref: 33_009)
- Contribution to SmartM2M TS 102 921: Correct re-targeting mechanism for the CoAP binding. 2015 (Ref: 33_010
- Contribution to OneM2M ARC TS-0001: Collection pattern for OneM2M. 2014 (Ref: ARC-2014-1553)
- Contribution to OneM2M ARC TS-0001: REST API versioning for OneM2M. 2014 (Ref: ARC-2014-1592)
- Contributions to OneM2M TST TS-0013: 2015 (Refs: TST-2015-0041R02, TST-2015-0042, TST-2015-0033R01, TST-2015-0034R01, TST-2015-0035R01, TST-2015-0031R02, and TST-2015-0032)

Bibliography

[1] D. Evans, "The internet of things: How the next evolution of the internet is changing everything," *CISCO white paper*, vol. 1, 2011.

[2] A. Bassi and G. Horn, "Internet of things in 2020: A roadmap for the future," *European Commission: Information Society and Media*, 2008.

[3] D. Boswarthick, O. Elloumi, and O. Hersent, *M2M communications: a systems approach*. John Wiley & Sons, 2012.

[4] E. Mainsah, "Autonomic computing: the next era of computing," *Electronics & Communication Engineering Journal*, vol. 14, no. 1, pp. 2–3, 2002.

[5] T. I. Association *et al.*, "White paper on cloud computing," 2009.

[6] J. Swetina, G. Lu, P. Jacobs, F. Ennesser, and J. Song, "Toward a standardized common m2m service layer platform: Introduction to onem2m," *Wireless Communications, IEEE*, vol. 21, no. 3, pp. 20–26, 2014.

[7] M. Brenner and M. Unmehopa, *The open mobile alliance: delivering service enablers for next-generation applications*. John Wiley & Sons, 2008.

[8] Z. Sheng, S. Yang, Y. Yu, A. Vasilakos, J. Mccann, and K. Leung, "A survey on the ietf protocol suite for the internet of things: Standards, challenges, and opportunities," *Wireless Communications, IEEE*, vol. 20, no. 6, pp. 91–98, 2013.

[9] A. Bassi, M. Bauer, M. Fiedler, T. Kramp, R. Van Kranenburg, S. Lange, and S. Meissner, *Enabling things to talk*. Springer, 2013.

[10] J. Soldatos, N. Kefalakis, M. Hauswirth, M. Serrano, J.-P. Calbimonte, M. Riahi, K. Aberer, P. P. Jayaraman, A. Zaslavsky, I. P. Žarko, *et al.*, "Openiot: Open source internet-of-things in the cloud," in *Interoperability and Open-Source Solutions for the Internet of Things*, pp. 13–25, Springer, 2015.

[11] E. Mingozzi, G. Tanganelli, C. Vallati, and V. Di Gregorio, "An open framework for accessing things as a service," in *Wireless Personal Multimedia Communications (WPMC), 2013 16th International Symposium on*, pp. 1–5, IEEE, 2013.

[12] J. Latvakoski, A. Iivari, P. Vitic, B. Jubeh, M. B. Alaya, T. Monteil, Y. Lopez, G. Talavera, J. Gonzalez, N. Granqvist, M. Kellil, H. Ganem, and T. VĄ0Ł1isĄ0Ł1nen, "A survey on m2m service networks," *Computers*, vol. 3, no. 4, pp. 130–173, 2014.

[13] M. B. Alaya, Y. Banouar, T. Monteil, C. Chassot, and K. Drira, "Om2m: Extensible etsi-compliant {M2M} service platform with self-configuration capability," *Procedia Computer Science*, vol. 32, no. 0, pp. 1079 – 1086, 2014. The 5th International Conference on Ambient Systems, Networks and Technologies (ANT-2014).

[14] E. Vogli, M. B. Alaya, T. Monteil, L. A. Grieco, and K. Drira, "An efficient resource naming for enabling constrained devices in smartm2m architecture," in *Proc. of IEEE International Conference on Industrial Technology*, (Seville, Spain), Mar. 2015.

[15] M. Ben Alaya, S. Medjiah, T. Monteil, and K. Drira, "Towards semantic data interoperability in onem2m standard," *IEEE Communications Magazine*, 2015.

[16] P. Horn, "Autonomic computing: Ibm\'s perspective on the state of information technology," 2001.

[17] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[18] IBM, "An architectural blueprint for autonomic computing," 2006.

[19] M. B. Alaya and T. Monteil, "FRAMESELF: an ontology-based framework for the self-management of machine-to-machine systems," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 6, pp. 1412–1426, 2015.

[20] M. B. Alaya and T. Monteil, "Frameself: an ontology-based framework for the self-management of machine-to-machine systems," *Procedia Computer Science*, vol. 27, no. 0, pp. 1079 – 1086, 2014. The 5th International Conference on Ambient Systems, Networks and Technologies (ANT-2014).

[21] M. B. Alaya, S. Matoussi, T. Monteil, and K. Drira, "Autonomic computing system for self-management of machine-to-machine networks," in *Proceedings of the 2012 International Workshop on Self-aware Internet of Things*, Self-IoT '12, (New York, NY, USA), pp. 25–30, ACM, 2012.

[22] M. Ben Alaya and T. Monteil, "Frameself: A generic context-aware autonomic framework for self-management of distributed systems," in *Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2012 IEEE 21st International Workshop on*, pp. 60–65, June 2012.

[23] J. Latvakoski, M. B. Alaya, H. Ganem, B. Jubeh, A. Iivari, J. Leguay, J. M. Bosch, and N. Granqvist, "Towards horizontal architecture for autonomic m2m service networks," *Future Internet*, vol. 6, no. 2, pp. 261–301, 2014.

[24] G. Gharbi, M. Ben Alaya, C. Diop, and E. Exposito, "Aoda: An autonomic and ontology-driven architecture for service-oriented and event-driven systems," vol. 3, pp. 167 – 188, 2013.

[25] G. Gharbi, M. Ben Alaya, C. Diop, and E. Exposito, "Aoda: An autonomic and ontology-driven architecture for service-oriented and event-driven systems," in *Proceedings of the 2012 IEEE 21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, WETICE '12, (Washington, DC, USA), pp. 72–77, IEEE Computer Society, 2012.

[26] K. Kim, S. Mehrotra, and N. Venkatasubramanian, "Farecast: fast, reliable application layer multicast for flash dissemination," in *Proceedings of the ACM/IFIP/USENIX 11th International Conference on Middleware*, Middleware '10, (Berlin, Heidelberg), pp. 169–190, Springer-Verlag, 2010.

[27] A. Aucinas, J. Crowcroft, and P. Hui, "Energy efficient mobile m2m communications," in *4th Extreme Conference on Communications (ExtremeCom2012)*, (Swiss Alps, Switzerland), March 2012.

[28] P. Bellavista, G. Cardone, A. Corradi, and L. Foschini, "Convergence of manet and wsn in iot urban scenarios," *Sensors Journal, IEEE*, vol. 13, no. 10, pp. 3558–3567, 2013.

[29] G. Tanganelli, E. Mingozzi, and C. Vallati, "A distributed architecture for discovery and access in the internet of things," in *IEEE INFOCOM*, (Turin, Italy), Apr. 2013.

[30] F. Andreini, F. Crisciani, C. Cicconetti, and R. Mambrini, "Context-aware location in the internet of things," in *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, pp. 300–304, 2010.

[31] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson, "Host Identity Protocol," April 2008. Request for Comments (RFC 5201).

[32] G. Xylomenos, C. Ververidis, V. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. Katsaros, and G. Polyzos, "A survey of information-centric networking research," *Communications Surveys Tutorials, IEEE*, 2013. To be published.

[33] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos, *et al.*, "Named data networking (ndn) project," *Relatório Técnico NDN-0001, Xerox Palo Alto Research Center-PARC*, 2010.

[34] L. Grieco, M. Ben Alaya, T. Monteil, and K. Drira, "Architecting information centric etsi-m2m systems," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on*, pp. 211–214, March 2014.

[35] L. Grieco, M. Ben Alaya, T. Monteil, and K. Drira, "A dynamic random graph model for diameter-constrained topologies in networked systems," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 61, pp. 982–986, Dec 2014.

[36] I. S. I. Group, "Internet of things (iot) and machine to machine communications (m2m)challenges and opportunities," *IoT SIG white paper*, 2013.

[37] M. Swan, "Sensor mania! the internet of things, wearable computing, objective metrics, and the quantified self 2.0," *Journal of Sensor and Actuator Networks*, vol. 1, no. 3, pp. 217–253, 2012.

[38] I. Ishaq, D. Carels, G. K. Teklemariam, J. Hoebeke, F. V. d. Abeele, E. D. Poorter, I. Moerman, and P. Demeester, "Ietf standardization in the field of the internet of things (iot): a survey," *Journal of Sensor and Actuator Networks*, vol. 2, no. 2, pp. 235–287, 2013.

[39] F. Ganji, E. M. Kluge, and B. Scholz-Reiter, "Bringing agents into application: intelligent products in autonomous logistics," in *Artificial Intelligence and Logistics (AILog)–Workshop at ECAI*, 2010.

[40] S. Haller, S. Karnouskos, and C. Schroth, "Future internet — fis 2008," ch. The Internet of Things in an Enterprise Context, pp. 14–28, Berlin, Heidelberg: Springer-Verlag, 2009.

[41] J. Zander and P. Mahonen, "Riding the data tsunami in the cloud: myths and challenges in future wireless access," *Communications Magazine, IEEE*, vol. 51, pp. 145–151, March 2013.

[42] I. Cha, Y. Shah, A. U. Schmidt, A. Leicher, and M. Meyerstein, "Security and trust for m2m communications 1," 2009.

[43] P. Clements, D. Garlan, L. Bass, J. Stafford, R. Nord, J. Ivers, and R. Little, *Documenting software architectures: views and beyond*. Pearson Education, 2002.

[44] M. Shaw and D. Garlan, *Software architecture: perspectives on an emerging discipline*, vol. 1. Prentice Hall Englewood Cliffs, 1996.

[45] R. A. Olsson and A. W. Keen, "Remote procedure call," *The JR Programming Language: Concurrent Programming in an Extended Java*, pp. 91–105, 2004.

[46] R. T. Fielding, *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine, 2000.

[47] S. Vinoski, "Rpc and rest: Dilemma, disruption, and displacement," *Internet Computing, IEEE*, vol. 12, pp. 92–95, Sept 2008.

[48] X. Feng, J. Shen, and Y. Fan, "Rest: An alternative to rpc for web services architecture," in *Future Information Networks, 2009. ICFIN 2009. First International Conference on*, pp. 7–10, IEEE, 2009.

[49] M. Maleshkova, C. Pedrinaci, and J. Domingue, "Investigating web apis on the world wide web," in *Web Services (ECOWS), 2010 IEEE 8th European Conference on*, pp. 107–114, IEEE, 2010.

[50] A. D. Birrell and B. J. Nelson, "Implementing remote procedure calls," *ACM Trans. Comput. Syst.*, vol. 2, pp. 39–59, Feb. 1984.

[51] T. Erl, *Service-oriented architecture: concepts, technology, and design*. Pearson Education India, 2006.

[52] L. Richardson and S. Ruby, *RESTful web services*. " O'Reilly Media, Inc.", 2008.

[53] L. Richardson, M. Amundsen, and S. Ruby, *RESTful Web APIs*. " O'Reilly Media, Inc.", 2013.

[54] T. ETSI, "102 690, machine-to-machine communications (m2m); functional architecture.," *European Telecommunications Standards Institute (ETSI)*, 2011.

[55] T. ETSI, "102 921, machine-to-machine communications (m2m); mia, dia and mid interfaces.," 2012.

[56] T. ETSI, "102 689, machine-to-machine communications; m2m service requirements.," *European Telecommunications Standards Institute (ETSI)*, 2010.

[57] A. Iivari, T. Vaisanen, M. Ben Alaya, T. Riipinen, and T. Monteil, "Harnessing xmpp for machine-to-machine communications and pervasive applications," *Journal of Communications Software and Systems*, vol. 10, no. 3, pp. 163 – 178, 2014.

[58] M. Salehie and L. Tahvildari, "Autonomic computing: emerging trends and open problems," in *ACM SIGSOFT Software Engineering Notes*, vol. 30, pp. 1–7, ACM, 2005.

[59] M. Parashar and S. Hariri, "Autonomic computing: An overview," in *Unconventional Programming Paradigms*, pp. 257–269, Springer, 2005.

[60] D. Garlan, S.-W. Cheng, A.-C. Huang, B. Schmerl, and P. Steenkiste, "Rainbow: Architecture-based self-adaptation with reusable infrastructure," *Computer*, vol. 37, no. 10, pp. 46–54, 2004.

[61] M. R. Nami and M. Sharifi, "A survey of autonomic computing systems," in *Intelligent Information Processing III*, pp. 101–110, Springer, 2007.

[62] D. Johnston-Watt, "Under new management," *ACM Queue*, vol. 4, no. 2, pp. 50–58, 2006.

[63] A. Khalid, M. A. Haye, M. J. Khan, and S. Shamail, "Survey of frameworks, architectures and techniques in autonomic computing," in *Autonomic and Autonomous Systems, 2009. ICAS'09. Fifth International Conference on*, pp. 220–225, IEEE, 2009.

[64] S. Hariri, B. Khargharia, H. Chen, J. Yang, Y. Zhang, M. Parashar, and H. Liu, "The autonomic computing paradigm," *Cluster Computing*, vol. 9, no. 1, pp. 5–17, 2006.

[65] A. J. Chakravarti, G. Baumgartner, and M. Lauria, "The organic grid: self-organizing computation on a peer-to-peer network," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 35, no. 3, pp. 373–384, 2005.

[66] K. Appleby, S. Fakhouri, L. Fong, G. Goldszmidt, M. Kalantar, S. Krishnakumar, D. P. Pazel, J. Pershing, and B. Rochwerger, "Oceano-sla based management of a computing utility," in *Integrated Network Management Proceedings, 2001 IEEE/IFIP International Symposium on*, pp. 855–868, IEEE, 2001.

[67] S. Rhea, C. Wells, P. Eaton, D. Geels, B. Zhao, H. Weatherspoon, and J. Kubiatowicz, "Maintenance-free global data storage," *Internet Computing, IEEE*, vol. 5, no. 5, pp. 40–49, 2001.

[68] G. M. Lohman and S. S. Lightstone, "Smart: making db2 (more) autonomic," in *Proceedings of the 28th international conference on Very Large Data Bases*, pp. 877–879, VLDB Endowment, 2002.

[69] S. Chaudhuri and V. Narasayya, "Self-tuning database systems: a decade of progress," in *Proceedings of the 33rd international conference on Very large data bases*, pp. 3–14, VLDB Endowment, 2007.

[70] D. M. Chess, A. Segal, I. Whalley, and S. R. White, "Unity: Experiences with a prototype autonomic computing system," in *Autonomic Computing, 2004. Proceedings. International Conference on*, pp. 140–147, IEEE, 2004.

[71] X. Dong, S. Hariri, L. Xue, H. Chen, M. Zhang, S. Pavuluri, and S. Rao, "Autonomia: an autonomic computing environment," in *Performance, Computing, and Communications Conference, 2003. Conference Proceedings of the 2003 IEEE International*, pp. 61–68, IEEE, 2003.

[72] H. Liu, M. Parashar, and S. Hariri, "A componentbased progarmming framework for autonomic applications," in *Proceedings of the 1st IEEE International Conference on Autonomic Computing*, 2004.

[73] G. Pujolle, "An autonomic-oriented architecture for the internet of things," in *Modern Computing, 2006. JVA'06. IEEE John Vincent Atanasoff 2006 International Symposium on*, pp. 163–168, IEEE, 2006.

[74] A. Ranganathan and R. H. Campbell, "Autonomic pervasive computing based on planning," in *Autonomic Computing, 2004. Proceedings. International Conference on*, pp. 80–87, IEEE, 2004.

[75] S. Abdelwahed, N. Kandasamy, and S. Neema, "Online control for self-management in computing systems," in *Real-Time and Embedded Technology and Applications Symposium, 2004. Proceedings. RTAS 2004. 10th IEEE*, pp. 368–375, IEEE, 2004.

[76] S. A. Gurguis and A. Zeid, "Towards autonomic web services: Achieving self-healing using web services," *ACM SIGSOFT Software Engineering Notes*, vol. 30, no. 4, pp. 1–5, 2005.

[77] G. Kaiser, J. Parekh, P. Gross, and G. Valetto, "Kinesthetics extreme: An external infrastructure for monitoring distributed legacy systems," in *Autonomic Computing Workshop. 2003. Proceedings of the*, pp. 22–30, IEEE, 2003.

[78] C. Anglano and S. Montani, "Achieving self-healing in autonomic software systems: a case-based reasoning approach.," *FRONTIERS IN ARTIFICIAL INTELLIGENCE AND APPLICATIONS*, vol. 135, p. 267, 2005.

[79] Z. Stanfel, Z. Hocenski, and G. Martinovic, "A self manageable rule driven enterprise application," in *29th International Conference on Information Technology Interfaces (ITI 2007)*, pp. 717–722, 2007.

[80] J. Appavoo, K. Hui, C. A. Soules, R. W. Wisniewski, D. M. Da Silva, O. Krieger, M. A. Auslander, D. Edelsohn, B. Gamsa, G. R. Ganger, *et al.*, "Enabling autonomic behavior in systems software with hot swapping," *IBM systems journal*, vol. 42, no. 1, pp. 60–76, 2003.

[81] E. Kiciman and Y.-M. Wang, "Discovering correctness constraints for self-management of system configuration," in *Autonomic Computing, 2004. Proceedings. International Conference on*, pp. 28–35, IEEE, 2004.

[82] M. Mesnier, E. Thereska, G. R. Ganger, D. Ellard, and M. Seltzer, "File classification in self-* storage systems," in *Autonomic Computing, 2004. Proceedings. International Conference on*, pp. 44–51, IEEE, 2004.

[83] M. J. Oudshoorn, M. M. Fuad, and D. Deb, "Towards autonomic computing: Injecting self-organizing and self-healing properties into java programs," *FRONTIERS IN ARTIFICIAL INTELLIGENCE AND APPLICATIONS*, vol. 147, p. 384, 2006.

[84] N. Gandhi, J. L. Hellerstein, S. Parekh, and D. M. Tilbury, "Managing the performance of lotus notes: A control theoretic approach," in *Proceedings of the Computer Measurement Group*, 2001.

[85] V. Markl, G. M. Lohman, and V. Raman, "Leo: An autonomic query optimizer for db2," *IBM Systems Journal*, vol. 42, no. 1, pp. 98–106, 2003.

[86] P. Shivam, S. Babu, and J. S. Chase, "Learning application models for utility resource planning," in *Autonomic Computing, 2006. ICAC'06. IEEE International Conference on*, pp. 255–264, IEEE, 2006.

[87] J. Park and P. Chandramohan, "Static vs. dynamic recovery models for survivable distributed systems," in *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on*, pp. 9–pp, IEEE, 2004.

[88] I. Rish, M. Brodie, N. Odintsova, S. Ma, and G. Grabarnik, "Real-time problem determination in distributed systems using active probing," in *Network Operations and Management Symposium, 2004. NOMS 2004. IEEE/IFIP*, vol. 1, pp. 133–146, IEEE, 2004.

[89] K. Mills, S. Rose, S. Quirolgico, M. Britton, and C. Tan, "An autonomic failure-detection algorithm," *ACM SIGSOFT Software Engineering Notes*, vol. 29, no. 1, pp. 79–83, 2004.

[90] P. Reynolds, C. E. Killian, J. L. Wiener, J. C. Mogul, M. A. Shah, and A. Vahdat, "Pip: Detecting the unexpected in distributed systems.," in *NSDI*, vol. 6, pp. 115–128, 2006.

[91] F. Qin, J. Tucek, J. Sundaresan, and Y. Zhou, "Rx: treating bugs as allergies—a safe method to survive software failures," in *ACM SIGOPS Operating Systems Review*, vol. 39, pp. 235–248, ACM, 2005.

[92] B. Claudel, N. De Palma, R. Lachaize, and D. Hagimont, "Self-protection for distributed component-based applications," in *Stabilization, Safety, and Security of Distributed Systems*, pp. 184–198, Springer, 2006.

[93] M. Costa, J. Crowcroft, M. Castro, A. Rowstron, L. Zhou, L. Zhang, and P. Barham, "Vigilante: End-to-end containment of internet worms," in *ACM SIGOPS Operating Systems Review*, vol. 39, pp. 133–147, ACM, 2005.

[94] M. Jarrett and R. Seviora, "Diversity to enhance autonomic computing self-protection," in *Availability, Reliability and Security, 2006. ARES 2006. The First International Conference on*, pp. 5 pp.–, April 2006.

[95] M. Bali, *Drools JBoss Rules 5.0 Developer's Guide*. Packt Publishing Ltd, 2009.

[96] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Middleware 2001* (R. Guerraoui, ed.), vol. 2218 of *Lecture Notes in Computer Science*, pp. 329–350, Springer Berlin Heidelberg, 2001.

[97] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," *Communications Surveys Tutorials, IEEE*, vol. 7, pp. 72–93, Second 2005.

[98] R. Jimenez, F. Osmani, and B. Knutsson, "Sub-Second lookups on a Large-Scale Kademlia-Based overlay," in *11th IEEE International Conference on Peer-to-Peer Computing 2011 (P2P'11)*, (Kyoto, Japan), Aug. 2011.

[99] S. A. Crosby and D. S. Wallach, "An Analysis of BitTorrent's Two Kademlia-Based DHTs," June 2007. Technical Report TR-07-04, Department of Computer Science, Rice University.

[100] A. Dhraief, A. Ghorbali, T. Bouali, A. Belghith, and K. Drira, "HBMON: A HIP based M2M overlay network," in *Network of the Future (NOF), 2012 Third International Conference on the*, pp. 1–8, 2012.

[101] A. Dhraief, M. Ghorbali, T. Bouali, and A. Belghith, "Mobility management in the HIP-based M2M overlay network," in *Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2013 IEEE 22nd International Workshop on*, pp. 50–55, 2013.

[102] A. Dhraief, A. Belghith, K. Drira, T. Bouali, and M. A. Ghorbali, "Autonomic management of the HIP-based M2M overlay network," *Procedia Computer Science*, vol. 19, no. 0, pp. 98 – 105, 2013.

[103] T. Vaisanen, "A simple m2m overlay entity discovery protocol," in *ICCGI 2012 : The Seventh International Multi-Conference on Computing in the Global Information Technology*, 2012.

[104] J. Latvakoski, T. Hautakoski, T. Väisänen, J. Toivonen, A. Lappalainen, and T. Aarnipuro, "Secure m2m service space in residential home," in *Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE*, COMSWARE '09, (New York, NY, USA), ACM, 2009.

[105] Y. J. Kim, E. Kim, B. W. Nam, and I. Chong, "Service composition using new dson platform architecture for m2m service," in *Information Networking (ICOIN), 2012 International Conference on*, pp. 114–119, 2012.

[106] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *ACM CoNEXT '09*, 2009.

[107] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan, "Building efficient wireless sensor networks with low-level naming," *SIGOPS Oper. Syst. Rev.*, vol. 35, pp. 146–159, Oct. 2001.

[108] S. Okamoto, N. Yamanaka, D. Matsubara, and H. Yabusaki, "Energy efficient and enhanced-type data-centric network (E3-DCN)," in *13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, 2012.

[109] G. Marias, N. Fotiou, and G. Polyzos, "Efficient information lookup for the internet of things," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012 IEEE International Symposium on a*, pp. 1–6, 2012.

[110] A. Rayes, M. Morrow, and D. Lake, "Internet of things implications on icn," in *Collaboration Technologies and Systems (CTS), 2012 International Conference on*, pp. 27–33, 2012.

[111] T. Biswasy, A. Chakrabortiz, R. Ravindranz, X. Zhangz, and G. Wangz, "Contextualized information-centric home network," in *In Proc. of ACM SIGCOMM*, (Hong Kong, China), Aug. 2013.

[112] R. Ravindran, T. Biswas, X. Zhang, A. Chakraborti, and G.-Q. Wang, "Information-centric networking based homenet," in *Proc. International Workshop on Management of the Future Internet (ManFI). IFIP/IEEE, 2013.*, 2013.

[113] J. M. Batalla, P. Krawiec, M. Gajewski, and K. Sienkiewicz, "Id layer for internet of things based on name-oriented networking," *Journal of Telecommunications and Information Technology (JTIT)*, vol. 2, pp. 40–48, 2013.

[114] M. Bari, S. Chowdhury, R. Ahmed, R. Boutaba, and B. Mathieu, "A survey of naming and routing in information-centric networks," *Communications Magazine, IEEE*, vol. 50, pp. 44–53, Dec. 2012.

[115] A. Hoque et al., "NLSR: Named-data Link State Routing Protocol," in *Proc. ACM SIGCOMM ICN Workshop*, (Hong Kong, China), Aug. 2013.

[116] M. Tortelli, L. A. Grieco, G. Boggia, and K. Pentikousis, "Cobra: Lean intra-domain routing in ndn," in *Special Session on Information Centric Networking, IEEE Consumer Communications and Networking Conference (CCNC)*, 2014.

[117] R. Chiocchetti, D. Perino, G. Carofiglio, D. Rossi, and G. Rossini, "INFORM: a dynamic INterest FORwarding Mechanism for Information Centric Networking," in *The 3rd ACM SIGCOMM Workshop on Information-Centric Networking (ICN)*, 2013.

[118] Y. Wang, K. Lee, B. Venkataraman, R. Shamanna, I. Rhee, and S. Yang, "Advertising cached contents in the control plane: Necessity and feasibility," in *In Proc. of IEEE INFOCOM WKSHPS*, (Orlando, (FL), USA), Mar. 2012.

[119] M. Lee, K. Cho, K. Park, T. Kwon, and Y. Choi, "SCAN: Scalable Content Routing for Content-Aware Networking.," in *In Proc. of IEEE ICC*, (Kyoto, Japan), Jun. 2011.

[120] H. Liu, X. Foy, and D. Zhang, "A multi-level DHT routing framework with aggregation," in *In Proc. of ACM ICN Workshop*, (Helsinki, Finland), Aug. 2012.

[121] H. Li et al, "Merts: A more efficient real-time traffic support scheme for content centric networking," in *ICCIT 2011*, 2011.

[122] D. Loguinov, J. Casas, and X. Wang, "Graph-theoretic analysis of structured peer-to-peer systems: Routing distances and fault resilience," *IEEE Transactions on Networking*, vol. 13, no. 5, 2005.

[123] P. Erdos and A. Renyi, "On the evolution of random graphs," *Mat Kutato Int. Kozl*, vol. 5, no. 17, pp. 17–60, 1960.

[124] S. Noel and S. Jajodia, "Understanding complex network attack graphs through clustered adjacency matrices," in *21st IEEE Annual Computer Security Applications Conference*, 2005.