Mobile Netw Appl (2014) 19:4–17
DOI 10.1007/s11036-013-0480-9

Connecting and Managing M2M Devices in the Future Internet

JaeSeung Song · Andreas Kunz · Mischa Schmidt · Piotr Szczytowski

Published online: 5 November 2013

Abstract As a cutting edge technology, Machine to Machine (M2M) communications are gaining ground for managing and controlling M2M devices. Since these objects have intelligence, communication capabilities and ability to work with their environment, they are considered as a key component for future smart buildings and cities. In this article we briefly introduce several architectures for M2M devices, which are being standardized in 3GPP, ETSI TC M2M and OneM2M, highlighting key functionalities used to manage and control M2M devices such as overload control, conflict management and semantic interworking. Further, the description of their realization in the field of smart building, as part of the CAMPUS 21 European project based on the M2M service modeling from the SENSEI European project, is presented.

Keywords Machine to machine · M2M · MTC · Standards · Smart City · Smart building · Smart object

J. Song
Sejong University, 98 Gunja-Dong, Gwangjin-Gu, 143-747, Seoul, Korea
e-mail: jssong@sejong.ac.kr

A. Kunz (✉) · M. Schmidt · P. Szczytowski
NEC Europe Ltd., Kurfürsten-Anlage 36, 69115 Heidelberg, Germany
e-mail: andreas.kunz@neclab.eu

M. Schmidt
e-mail: mischa.schmidt@neclab.eu

P. Szczytowski
e-mail: Piotr.Szczytowski@neclab.eu

1 Introduction

The Internet of Things (IoT) is becoming more and more popular for vertical industries and the deployment of systems providing services for M2M devices for different tasks is already ongoing in many countries. These systems face several issues in providing a harmonized service for the M2M application provider, since they are highly specialized and isolated in silos [2, 14, 18].

In particular, managing and controlling M2M devices in M2M systems usually pose a number of challenges:

– *Identification and discovery*: Within the end to end M2M framework, all individual entities including physical sensors and software M2M applications must be uniquely identifiable. The M2M system should also be flexible in a way supporting identification and discovery of group of devices or virtual devices by not only using identifier but also many other means such as location, pseudo-name or combination thereof.
– *Overload*: Although the huge number of smart devices is expected to be connected to networks, existing wireless and wired network protocols were not developed by considering their special needs. Therefore, optimizing currently deployed network infrastructures to cope with this problem is an important issue to be tackled.
– *Resource management*: There exist many factors that make it difficult for the M2M system to manage its resources. They include: the complexity and variability of M2M system environments; the scale of the network size; complicated operation status; many constrained resources.
– *Semantic interoperability*: In heterogeneous environments where different types of devices and services

communicate with each other, rapid and flexible information exchange is required. However, these devices and services often use different data models, they are not interoperable and dont understand each other. Semantic information must be extracted from physical objects and expressed in a meaningful way for machines.
- *Conflict management*: M2M applications control various devices and often their policies for controlling devices cause conflicts. Conflicts arise when multiple M2M applications try to control the same actuator in different ways, or when the consequences of a control interfere with other M2M applications. The proper resolution is necessary to the success of a M2M system

In this paper, we first cover several standardization activities of the European Telecommunications Standards Institute (ETSI),[1] 3rd Generation Partnership Project (3GPP)[2] and one Machine to Machine (oneM2M),[3] focusing on those related to manage and control M2M devices. We will present current available solutions specified in the ETSI M2M and 3GPP Machine Type Communication (MTC) architectures. We then introduce how these existing solutions can be enhanced and applied to a real world application domain.

The 3GPP architecture for MTC focuses on managing signalling overload from M2M devices while the ETSI M2M architecture provides a complete end-to-end M2M service platform together with an intermediate service layer which intends to support multiple vertical industries such as eHealth and smart metering. These two architectures provide features that handle some challenges such as overload control, resource identifiers and discovery mechanisms. Additionally, we provide an introduction of a new M2M standard initiative called oneM2M and their potential layered architecture since most standard activities related to M2M will be moved to this new initiative. A separate section (Section 3) addresses key technologies of M2M system that provide solutions for the addressed challenges together with their limitations. The two European projects, SENSEI[4] [16] and CAMPUS 21 [3] are described together with several potential enhancements. In addition, we introduced a prototype implementation showing the feasibility and proof of concept of CAMPUS 21.

The remainder of this paper is organized as follows: Section 2 provides standardized architectures for M2M devices. In Section 3, a set of key technologies are presented together with some potential enhancements followed in Section 4 by introducing a semantic model and architecture which can provide the energy-efficient operation for smart buildings. A set of conclusions and further steps are described in Section 5.

2 Architectures for M2M devices

Since M2M devices are linked and managed through communication technologies which require an integration and convergence between many different communication systems, architectures for M2M are developed in different standards organizations. Therefore, in this section we introduce M2M architectures standardized in ETSI TC M2M, 3GPP and oneM2M [4].

2.1 ETSI TC M2M

ETSI has formed a Technical Committee (TC) in January 2009 to perform standard work dedicated to M2M. The main goals of the committee are developing and maintaining an end-to-end architecture for M2M systems together with providing features, such as sensor network integration, QoS, security, discovery, location, etc. These are common features that can be used to manage M2M devices across various applications from smart building to city automation.

The ETSI M2M architecture is composed of three domains, *M2M area network*, *communications network* and *application domain*, as shown in Fig. 1. M2M components embedded in a M2M device are deployed in the M2M area network domain and reply to requests for information or transmit data automatically. Examples of these components include temperature sensors, actuators, humidity sensors, fire detectors, etc. M2M gateways provide connectivity between M2M components and deliver collected data to communications network. The M2M area network, which links M2M components and M2M gateways, can be realized as various location area networks such as ZigBee,[5] Power Line Communication (PLC),[6] wireless M-BUS,[7] Bluetooth[8] or IEEE 802.15.[9] The communications network enables connectivity between M2M gateways and M2M application servers. This connectivity can be provided by various access networks such as cellular network, fixed network and satellites.

[1]http://www.etsi.org/
[2]http://www.3gpp.org/
[3]http://www.onem2m.org/
[4]http://www.sensei-project.eu/
[5]http://www.zibgee.org/
[6]http://www.hd-plc.org/
[7]http://www.m-bus.com/
[8]http://www.bluetooth.com/
[9]http://www.ieee802.org/15/

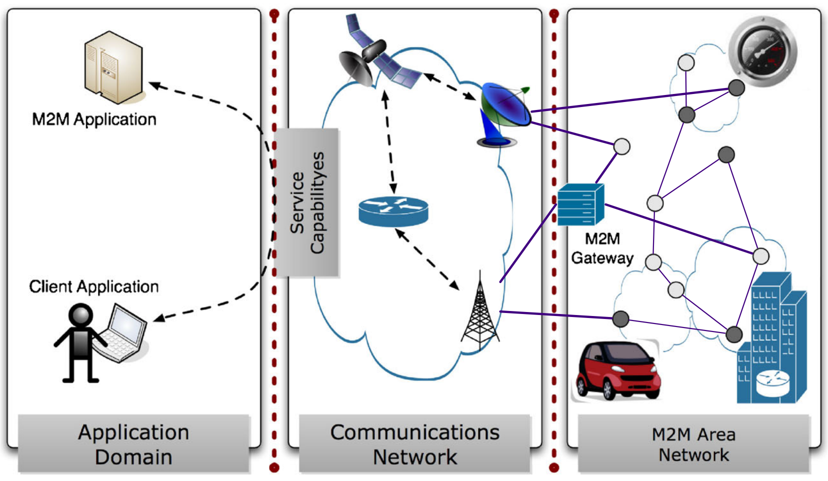


Fig. 1 Generic ETSI TC M2M architecture

2.2 3GPP machine type communications

The architecture defined in 3GPP [24] provides the optimized transport, subscriber management and triggering of M2M devices using 3GPP access technologies like UMTS or LTE. The architecture was developed based on requirements mainly received from ETSI M2M, who needed e.g. a trigger to wake up M2M devices. The terminology was slightly different to ETSI and called Machine Type Communication (MTC) and was introduced in 3GPP's Release 10. There are no specific MTC devices in 3GPP, the new functionalities and enhancements for MTC are also applicable to normal mobile devices, also called User Equipment (UE). Due to the huge amount of work, specification activities are still ongoing in its current Release 12 [20]. A good overview of all features in 3GPP can be found in [10]. While the MTC features of Release 10 did not require changes in the architecture [23], the newly introduced device triggering procedures showed the need of new network entities and interfaces to existing ones. The following Fig. 2 shows the Release 11 network architecture for MTC [24]:

The communication is carried out between the MTC Application in the User Equipment (UE) and the MTC Application Server (AS), which resides outside of the mobile operator domain. As highlighted in the figure, there are two nodes newly introduced to support the communication between UE and AS:

Services capability server (SCS) – The SCS connects the MTC AS to the 3GPP network and can be controlled by either the mobile operator or the MTC Service Provider, who is hosting the MTC Applications. The UE can also host one or more MTC Applications and the SCS is responsible to provide the relevant capabilities for those MTC Applications. 3GPP does not describe the mapping of MTC capabilities to the mobile operator network services, i.e. the reference point between SCS and MTC AS is out of scope.

MTC interworking function (MTC-IWF) – This is the major control node for MTC in the home network, interfacing all relevant network nodes. Its main functionality is related to identification and triggering of MTC devices, e.g. it authorizes the SCS and receives trigger requests from the SCS, does necessary protocol translation and then maps the E.164 MSISDN or External Identifier to International Mobile Subscriber Identity (IMSI) and selects the best device trigger delivery mechanism.

The MTC architecture allows three different ways to connect the MTC Application Server to the 3GPP network:

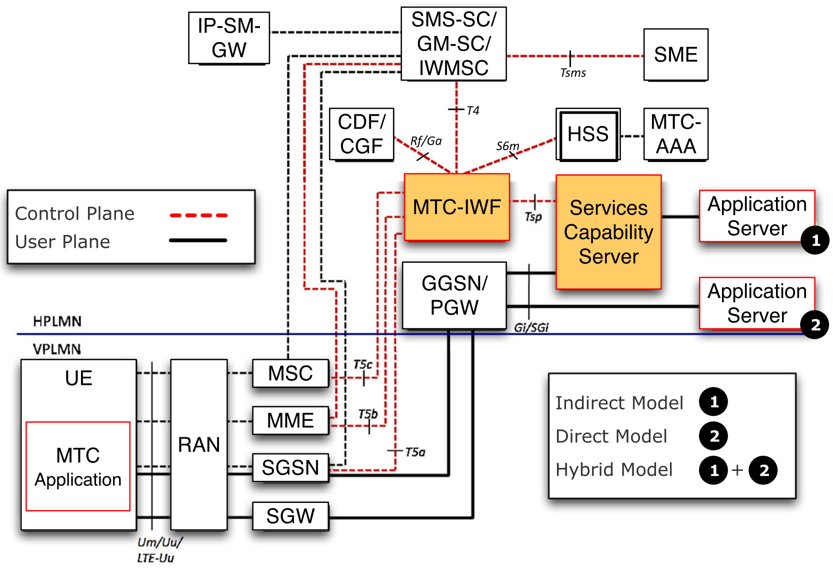
Indirect model – The MTC AS is connected indirectly to the mobile operator network via the SCS for utilization of additional value added services for MTC e.g. control plane device triggering. The SCS can be either MTC Service Provider controlled or 3GPP network operator controlled.

Direct model – In the direct model, the MTC AS is connected directly to the operator network without any SCS, enabling user plane communication between MTC AS and MTC UE.

Hybrid model – The MTC AS can use the direct model and indirect models simultaneously. There is no correlation between the direct user plane communication from the AS and any value added control plane related communications from the SCS.

3GPP has added two features, *device triggering* and *PS-only service provision*, in Release 11 specifications. *Device triggering* is a feature that enables network entities to send a trigger indication message to the UE. When the UE should perform application specific actions with the SCS (indirect model) or the MTC AS (hybrid model) and the SCS/MTC AS do not know or cannot reach the IP address of the MTC UE, then device triggering is performed. The trigger message is basically a Short Message (SMS) routed to

Fig. 2 3GPP MTC architecture



the destination UE and the corresponding application [22]. The UE is able to distinguish a device trigger message from other type of messages e.g. based on the SMS Application Port ID. The payload of the trigger message may have information for the application in the UE that triggers specific actions. Whether a UE can be triggered is based on the subscription profile. The trigger message may be routed via MSC, SGSN and/or MME and even via IMS. The delivery of SMS in MME is a new feature for the MME. *PS-only service provision* is a subscription where the UE uses exclusively the PS domain for all services, i.e. packet bearer services and SMS services. Only SMS services may be provided to the UE via the CS domain when the serving node or network does not support SMS via PS domain NAS.

2.3 oneM2M

Today many industries are addressing M2M solutions and their applications that often need customized hardware and software, which are typically difficult to deliver on time to market and cause higher CAPEX as well as higher OPEX. Thus there is a strong need to develop a horizontal common platform across a number of industry verticals, which can increase the efficient development of M2M systems.

A consortium consisting of seven standards development organizations (SDOs), i.e., ARIB, ATIS, TTA, CCSA, ETSI, TTC and TIA, has initiated a new global standard organization called oneM2M to work on developing technical specifications to provide a common M2M service layer. The common layer is intended to be embedded within various hardware and software components to ensure M2M devices can communicate on a global scale.

The main objective of oneM2M is to develop one globally agreed M2M specification. For this, oneM2M plans to consolidate current M2M service layer standards activities and collaborate with various SDOs and industry fora in charge of developing vertical market aspects of M2M applications. All these activities will be performed with the aim of lower operational costs, shorter time-to-market, avoidance of standardization overlap and realization of mass market economies of scale.

Figure 3 shows the generic architecture of oneM2M. The architecture has two common building blocks (i.e., common M2M service layer and common features for verticals) to serve the purpose of common M2M services where oneM2M will initially focus on creating its specifications. The features of these two building blocks are listed briefly in the following:

- Protocols/APIs/standard objects based on this architecture (open interfaces & protocols)
- Security and privacy aspects (authentication, encryption, integrity verification)

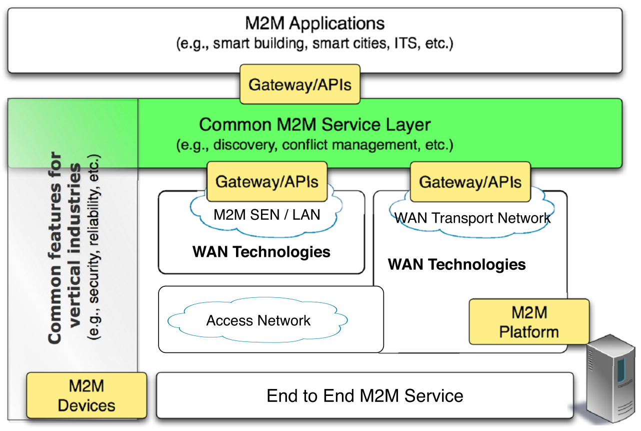


Fig. 3 Potential oneM2M layered architecture

- Reachability, discovery, identification and naming of devices and applications
- Charging aspects (charging data, not billing)
- Information models (including semantic) and data management
- Management aspects (including remote management of entities)

3 Key technologies in management and control

In this section, we present the key technologies that can be used to manage and control M2M devices.

3.1 Identification and discovery

The function of device identification is to assign the unique identifier to an entity while discovery is to find a desired entity using a given identifier. Since a huge number of M2M devices (more than 50 billion according to some forecasts in the next decade) are expected in the future, well defined M2M identification and discovery functions are a crucial part of the overall M2M system.

3GPP identifier 3GPP defines two identifiers for M2M devices – an external and an internal identifier. The internal identifier (i.e. IMSI) is defined to identify M2M devices within the 3GPP core network. On the other hand, the external identifier is required for a M2M device to be identified from an external network perspective and shall be globally unique. The external identifier is composed of two parts, one is to identify the domain of the network operator (domain identifier) and the other is flexibly allocated within the operator domain (local identifier). 3GPP allows assigning multiple external identifiers to one internal identifier and their relationships are stored in the operators subscriber database, the Home Subscriber Server (HSS) .

ETSI identifiers Since ETSI M2M specifies the end to end service platform for M2M, a complete set of identifiers are defined in the ETSI specifications. Some examples of identifiers include Application identifier (App-ID), designed to identify M2M applications, Service Capability Layer identifier (SCL-ID), used to identify service capability layers in the M2M Device/Gateway, M2M node identifier (M2M-Node-ID) for identifying M2M node, M2M Service Connection identifier (M2M-Connection-ID), defined to identify the connection between the M2M Device/Gateway and the Network and M2M Service Provider identifier (M2M-SP-ID) to uniquely identify M2M service provider.

Device and service discover Within M2M networks, services and devices are dynamically changed over time, and this nature makes it difficult to discover proper devices that must be required to use a particular service. One form of device discovery is provided through the use of resource directory that manages all the resources including devices and services in the network. In this case, each device or service must be registered with the resource directory in order to be discoverable. The resource directory maintains the information of registered resources and is used for resource lookup by resource users. This mechanism is similar to that used in GSM, i.e. the Home Location Register, which is a database to store and manage information of the current attachment of a mobile phone such as identifier and location. M2M systems can also use a distributed way of device discovery mechanism. For this, a device, which is added to a network and wants to be discovered by others, needs to advertise itself to the network, which is usually called *device announcement*. This introduces a certain degree of semantics for the network and therefore the user, who wants to discover a device, is able to find the desired device without having complex queries. In the ETSI M2M system, an announced device consists of a limited set of attributes, such as search strings (e.g. `searchStrings`) and the link to the original resource (i.e. `URI`), and the discovery procedure allows to retrieve the list of URIs of devices matching a discovery filter criteria. Universal Plug and Play (UPnP) [9][10] and Zeroconf [17], which are used as M2M enabling technologies, also adopt this mechanism for the device discovery.

3.2 Overload control in 3GPP

With a large number of M2M devices in the network, overload and congestion could easily occur when many of the devices want to transmit data at the same time and ask for resources in the network or when many of those M2M devices are roaming but the serving network fails so that all move onto local networks and suddenly overload them. In order to handle such scenarios, overload control was one of the first M2M features introduced in Release 10 generically and not limited to M2M [21, 23]. The basic principles of overload control consist of rejecting messages, using back-off timers in the M2M devices as well as long timers for mobility management and prioritization. If a M2M device is configured with low access priority, then already the Radio Resource Control (RRC) connection establishment requests can be rejected by the Radio Access Network (RAN) in overload situations. The RAN can additionally provide an extended wait timer to the M2M device, so that it has to wait until it can send the next RRC request.

Furthermore the mobility management timer for periodic Tracking Area Updates (TAU) can be extended so that the

[10]http://www.upnp.org/

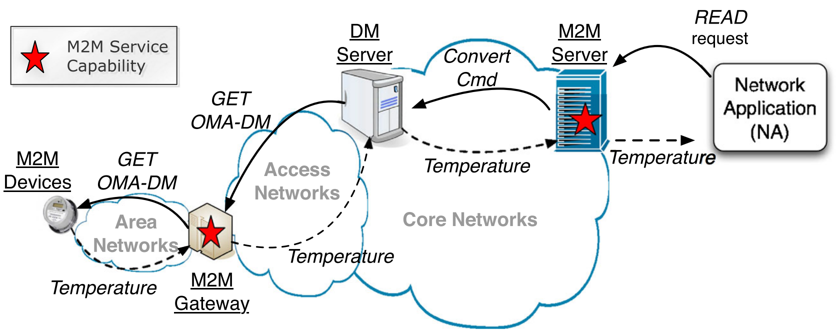


Fig. 4 An example of M2M device management

M2M device does not immediately recognize a network failure and would move later to a competing local network. Additional to this, the minimum periodic PLMN search time limit can be extended too. The Downlink Data Notification requests can be limited by the Mobility Management Entity (MME) for low priority traffic, i.e. M2M devices in idle mode would not be paged to wake up and receive data. The MME can apply on a per UE and per Access Point Name (APN) granularity a Non Access Stratum (NAS) level congestion control for mobility management (EMM) and session management (ESM) signaling messages, i.e. in overload situations, the MME can provide the M2M device with a back-off timer and the MME will reject all EMM and ESM requests from that device until the timer is expired. Another possibility to reduce the signaling load in the network for M2M devices was proposed in [18] using bulk signaling, i.e. aggregating common header of specific messages.

In order to support the methods described above, the M2M device needs to be configured to support them, too. The following options were specified in [23] and of course multiple configurations can be supported by the UE:

- low access priority; and/or
- perform Attach with IMSI at Public Land Mobile Network (PLMN) change; and/or
- long minimum periodic PLMN search time limit; and/or
- handling of the invalid Universal Subscriber Identity Module (USIM) state, the “forbidden PLMN list”, the “forbidden PLMNs for attach in S1mode list” and the “forbidden PLMNs for GPRS service list”; and/or
- Extended Access Barring (EAB).

The low access priority requires the support of EAB and vice versa and may require longer back-off timers, i.e. the device should be delay tolerant for accessing the network.

3.3 Resource management

M2M systems need to manage networks having billions of resources including software, hardware and objects. M2M resource management facilitates device management, configuration management, performance management, etc. and it supports various types of M2M devices from composited devices to constrained M2M devices. M2M resource management involves remote device management, device monitoring and device control. For example, a M2M application can trigger a device to activate and send an instruction to the device in order to set up measurement intervals. The application can also send a command to an actuator to perform a certain task such as firing an alarm.

In some particular application domains such as Smart buildings, M2M resource management extends its scope to even energy management. Energy management functions are used for lighting, heating, controlling air conditioning and then detecting empty rooms, etc.

For resource management, ETSI TC M2M has adopted a RESTful style software architecture [15]. In this architecture style, the information is represented by resources, which are structured as a tree, and exchanged between M2M Applications (DA, GA, NA) and M2M SCL. All supported functionalities such as registrations and device management actions are handled by the four basic operations CRUD, i.e. `Create`, `Retrieve`, `Update` and `Delete`, together with two additional methods – `Notify` and `Execute`. The ETSI M2M system also provides a service capability called a Remote Entity Management (xREM). In the ETSI M2M architecture, the REM capability is located in three places – the M2M Network Core (NREM), the M2M Gateway (GREM) and the M2M Device (DREM). The xREM capability provides several functionalities such as configuration and performance management, connection establishment triggering, software and firmware upgrade, etc.

Figure 4 shows the locations of each REM capability and their relationship. The NREM is managing the DREM and GREM. Existing protocols, i.e., the Device Management (DM) from Open Mobile Alliance (OMA)[11] and TR069 [19] from BroadBand Forum (BBF),[12] are adopted to support the interactions between the NREM and DREM/GREM. Since OMA-DM and BBR-TR069 do not use standardized RESTful management commands such

[11] http://www.openmobilealliance.org/
[12] http://www.broadband-forum.org/

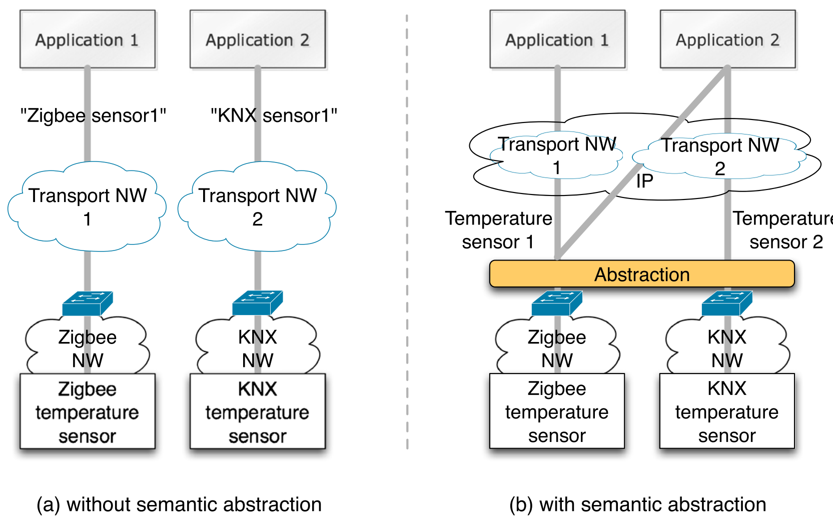


Fig. 5 The meaning of M2M devices in a system **(a)** without semantic abstraction and **(b)** with semantic abstraction

as Remote Procedure Calls (RPC), the non-RESTful commands are modeled as RESTful operations at the M2M Network Core.

For example, if a Network Application (NA) issues a `READ` request on a temperature sensor resource to the M2M Core via the mIa interface between NA and Service Capabilities in the M2M Network Domain (NSCL). The `READ` request is then processed by NREM, which will convert the RESTful command to a `GET` OMA-DM command for OMA-DM or a `GetParameterValues` RPC method for BBF-TR069. The result temperature is then delivered to the NREM followed by being forwarded to the NA.

3.4 Semantic interworking

In the context of the M2M system such as smart buildings or smart cities, devices are typically connected to different access networks, e.g. 3G , 4G, ZigBee, Bluetooth, and using different communication protocols for exchanging data, e.g. UPnP and X10 [5]. Enabling communications between these heterogeneous devices using different data format and protocols are hard to achieve.

Semantic interworking is being used to help resources and services in M2M systems to exchange information with one another. Semantic interworking ensures that these exchanges work properly that they have a common understanding of the meaning of the exchanged information. Semantic information collected from different sensors and occurrences can be analyzed and used to derive information that gives better understanding about real environment and to create more value added M2M applications. For example, reading temperature sensors and detecting the occupancy of rooms can be used to manage power consumptions in a building; monitoring and analyzing road traffics can provide smart city management such as smart traffic signaling systems.

To support this, ETSI M2M has started a new work item, “Study on Semantic support for M2M Data”, that aims to study a new Service Capability. The work item is investigating abstraction, discovery, interpretation and the use of the M2M data from different sources, without any kind of prior knowledge of that. This is essential to provide M2M horizontal services and to develop new open business opportunities using semantic M2M data. Devices defined by different vertical standards are abstracted into semantic information and considered semantically equivalent. For instance, as shown in Fig. 5 (left), two temperature sensors, one is defined in ZigBee and the other in KNX,[13] are specified differently since each standard has their own resource structure. Therefore, a typical M2M system understands both sensors as “ZigBee temperature sensor” and “KNX temperature sensor”. However, if they are abstracted semantically such as Fig. 5 (right), both sensors are semantically equivalent and delivering the same information “temperature sensor” to NA. In addition, discovery mechanisms for semantic information, handling of different types and amount of information that is available in the M2M system, considerations of ownership/responsibility for M2M resources are currently being standardized.

At the time of writing this article, the ETSI TC M2M working group has been developing use cases and high level architecture for semantic interworking within the work item. However, since we believe that semantic interworking plays a crucial role in the future M2M platform managing M2M devices, we have investigated how heterogeneous devices and services can be abstracted in an EU project which will be introduced in Section 4.1.

[13] http://www.knx.org/

3.5 Conflict management

As M2M applications may control various devices, it is clear that there is the potential for conflicts between controls. These conflicts have been resolved manually by human, however M2M systems need to detect and resolve them appropriately without human interventions.

For example, as shown in Fig. 6, let us consider that two M2M devices are deployed at home, one for a fire control and the other for a flood control. The fire control device detects a fire at home and sends a signal to an M2M application for the fire system. Then the application sends a signal to turn on some sprinklers during the fire. On the other hand, turning the sprinklers can cause flood situation at the basement before the fire is under control and this will make the flood control device to send a signal to an M2M application to the flood control system. The flood control application then sends a command to shut off the home's water main, rendering the sprinklers useless. As a result, the house will burn down further.

Tackling conflicts between controls requires several enhancements in the M2M device management function. First, conflicts must be detected. The purposes of conflict detection are to identify actual conflicts that have occurred at runtime; to predict a potential conflict which may occur in the future; and to communicate with associated M2M entities to resolve occurred or predicted conflicts. Second, once any conflicts are detected, they must be resolved. *Conflict resolution* is to determine when it is adequate to resolve the conflict and to analyze how to resolve the conflict using an appropriate method.

There already exist some work that has been attempted to detect and resolve conflicts in specific domains [6–8]. An interesting research is a work performed by Dunlop et al. [7, 8] that proposes a policy model and methods for a conflict detection and resolution in policy-based management systems. Required solutions in the domain of M2M to some extent have similarities to their methods. However, the environment of their target systems is substantially different from the M2M systems which have more things to be considered such as commands, services and various types of end nodes. The work from Cheng et al. [6] uses process specification language to exchange project scheduling information in the constuction industry. The exchanged scheduling information is then used to detect and coordinate conflicts among different applications based on the four dependency relationships: *Finish to Start*, *Finish to Finish*, *Start to Start* and *Start to Finish*. These four relationships could be applied to M2M systems for resolving detected conflicts between multiple commands to the same M2M resource. However, the relationships are limited to resolve a conflict ocurring in a single M2M resource while The environment of M2M systems are more pervacive, and conflicts are often ocurring among multiple resources. These differences and limitations of the previous work are then need to be considered in designing and developing conflict resolution techniques in M2M systems.

More technically, M2M applications associated with a conflict need to use at least some of the following functions to resolve the conflict:

- Service configuration and session re-establishments;
- Remote monitoring and controlling supports (i.e. check device status and instruct configuration);
- Remote upgrade for software, firmware and configuration;
- Diagnostics and monitoring network information such as connectivity and latency

Although conflict management is an important issue for the success deployment of the M2M system, this has not yet been addressed in any standard specifications related to M2M.

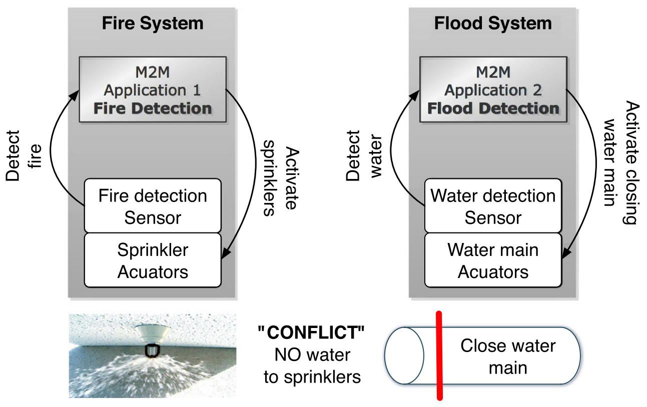


Fig. 6 Conflict between Fire and Flood M2M systems

4 Smart building system realization

One particular M2M application domain with relevance to aim of optimizing energy usage and carbon emissions is the area of smart buildings. Building Management System (BMS) Technologies for automated and centralized control exist in the industry since decades yet usually the various different system domains within a building such as Heating Ventilation and Air Conditioning (HVAC), electricity and water usage usually were not integrated. Realizing the lack of integration across the different domains it is not surprising that often in current buildings suboptimal configurations exist with contradicting settings. Further, given the lifespan of buildings and their built-in systems, the landscape of

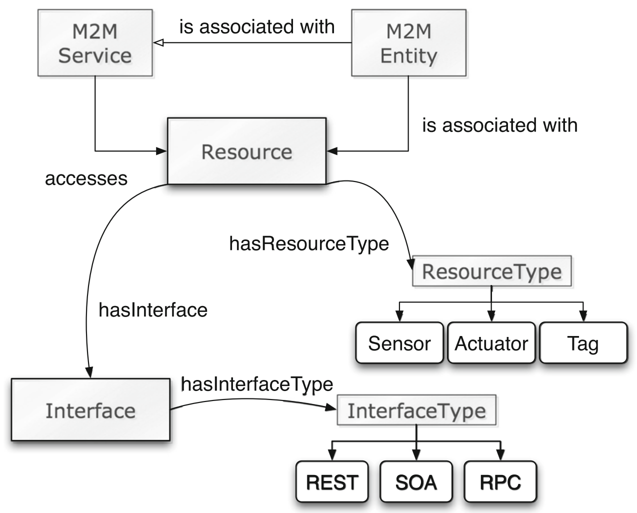


Fig. 7 SENSEI resource model

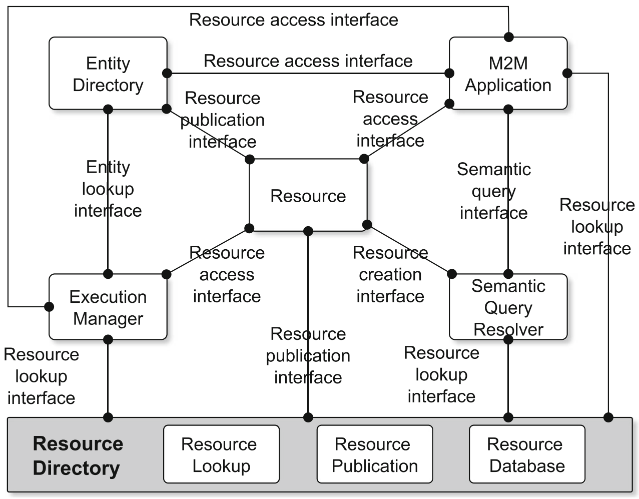

Resource access interface
Entity Directory
Resource access interface
M2M Application
Resource publication interface
Resource access interface
Entity lookup interface
Resource
Semantic query interface
Resource lookup interface
Resource access interface
Resource creation interface
Execution Manager
Semantic Query Resolver
Resource lookup interface
Resource publication interface
Resource lookup interface
Resource Directory
Resource Lookup
Resource Publication
Resource Database

Fig. 8 SENSEI generic architecture

technologies used is very diverse: Standardized technologies such as KNX, BACnet,[14] M-BUS, OPC[15] and many proprietary technologies are in use in today's buildings.

For fostering the research managing and controlling M2M devices, we developed a resource model that provides an abstraction of real entities so that various M2M applications can have common views regardless of used technologies (see Section 4.1). Based on the derived resource model, we are now developing a M2M system that provides the control and automated management of buildings (see Section 4.2).

4.1 SENSEI

The SENSEI project [16] aimed to realize smart environment for the Real World Internet and developed a global framework of common interfaces for sensor and actuator. A resource model is the core of SENSEI and it represents an entity in the world of the IP network.

Figure 7 introduces the resource model used in SENSEI [25]. Within the resource model, all things in the physical world such as sensors and actuators are described as "Entity". The actual software component that provides information on the associated entity and enables an access to control the actual device is specified as "Resource". A "Service" can have an access to an interesting resource through a well-defined and standardized "Interface" which offers various protocols such as REST and RPC.

The fundamental of SENSEI is to model all M2M devices such as sensors, actuators and processors as *Resources* as shown in Fig. 8. Based on the resource model, the SENSEI architecture supports various features such as resource functionalities, access and management methods in an abstracted view. They can be used to solve several challenges that we have addressed in Section 1. In this project, we specified this underlying resource data model for context information and actuation together with standardized interfaces. We take this resource model as an input for developing a new M2M platform which will be introduced in Section 4.3.

Resource identification and discovery There source description is derived from the concrete instantiation of this information and is published in the Resource Directory (RD) that acts as a service directory and implements a rendezvous functionality. For the purpose of discovery, resources are identified by unique identifiers in the SENSEI network and are described by a number of search terminologies.

Semantic interworking In order to provide a common view to resources in the real world, SENSEI introduces abstraction levels. In the lower level, all resources are accessible based on a common way of describing resources. A higher abstraction is to enable the development of M2M applications that can easily discover and use resources in different environments. This hides the heterogeneity of underlying infrastructures such as sensors using different access technologies. For this purpose, SENSEI proposes an advanced resource description to capture the syntax and semantics of the resource interface and the description provides an ontology including information about location, resource type and supported operations of a resource. The Entity Directory provides the link between the lower level and the higher level so that given a requested identifier of the higher level entity can be mapped to the value of an attribute in the lower level resources.

[14]http://www.bacnet.org/

[15]http://www.opcfoundation.org/

Conflict management The Semantic Query Resolver (SQR) provides a common interface to the M2M applications for all their requests. Requests from an application for an interesting resource are directed to SQR which creates an operation plan. The plan is then forwarded to the Execution Manager that analyzes the execution and the result is provided to the M2M application. In order to detect conflicts, the operation specifies the inputs that a resource can take and a corresponding output. The pre-conditions and post-conditions are also described in each operation so that the SENSEI framework can detect and resolve conflicts between operations.

Discussions and limitations The resource model developed in the SENSEI project provides an abstraction layer which hides the different underlying technologies and enables M2M applications to view M2M devices in a uniform manner. The SENSEI architecture also provides several advanced functionalities such as conflict management and semantic based resource discovery. However, SENSEI does not provide any enhancements on the aspects of managing and controlling the large scale M2M devices. SENSEI developed the Open M2M Application API together with EURESCOM study P1957 [13] and has submitted the OPEN API to ETSI M2M for standardization in 2011. However, M2M related standardizations such as ETSI M2M, oneM2M, 3GPP MTC and OMA NSGI are at an early or mid stage at the moment and therefore are not yet suitable for many use cases.

4.2 CAMPUS 21

Compared to the SENSEI project, the EU FP7 project "Control and Automation Management of Buildings and Public Spaces" (CAMPUS 21) focuses more on the energy-efficient operation & management of public buildings and spaces. It brings together research centers, Public Authorities and multiple industry sectors, such as Construction & Facilities Management, Building Services Systems Manufacturers and Energy Providers. The project looks into business model and procurement schemes as well as developing a technical solution to address holistic energy optimization for given buildings. For this purpose a variety of demonstration sites (a living lab, a municipal sports centre and a football stadium) will be used to verify and demonstrate optimization potentials. One particular challenge of the project is to integrate different BMS technologies, additional in-building systems (e.g., ticketing or scheduling systems) and micro energy generation systems (such as solar panels) in order to enhance the knowledge base about actual building use and optimize the energy flow both in terms of consumption as well as with respect to smart grids.

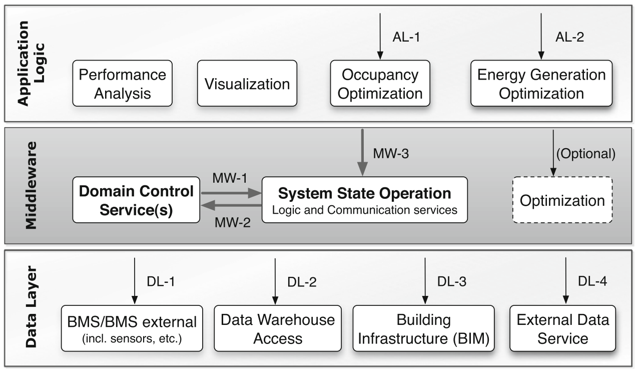


Fig. 9 CAMPUS 21 architecture

Architecture CAMPUS 21 developed a system architecture shown in Fig. 9 [3] reusing established communication technologies to integrate the diverse set of existing in-building systems of the different demonstration sites. For this it uses standard technologies such as BACnet to integrate either with an existing BMS or directly interact with the in-building sensing and actuation systems. The middleware layer Domain Control Service(s) component collects data from the different building and feeds it into a Data Warehouse for analysis by Optimizers sitting in an Application Layer on top of the middleware. The System State Operator component orchestrates access requests from the application logic layer (e.g. for data analysis, optimization or visualization) to the data layer (e.g. external ICT systems for ticketing, existing in-building BMS components or the Data Warehouse).

In CAMPUS 21, we are developing the core M2M middleware platform integrating the information from different building management subsystems such as BMS, monitoring various resources and external context systems.

For implementing the middleware, we chose the established Open Services Gateway initiative (OSGi) technology standards with the motivation that potentially, the in-building components of the CAMPUS 21 middleware can run on embedded devices with a small footprint.

At time of writing, the implementation of the middleware and the optimizers as well as the integration of the CAMPUS 21 system with the different demonstration sites is still ongoing. In the next section, we introduce a prototype implementation of the M2M middleware that we designed in CAMPUS 21.

4.3 Prototype implementation – ISIS

Based on the two EU projects, we have been developing a prototype M2M platform called ISIS (integrated M2M Platform). The platform is a middleware that provides the functionality of the collection, reasoning and distribution of M2M information from M2M devices and addresses three

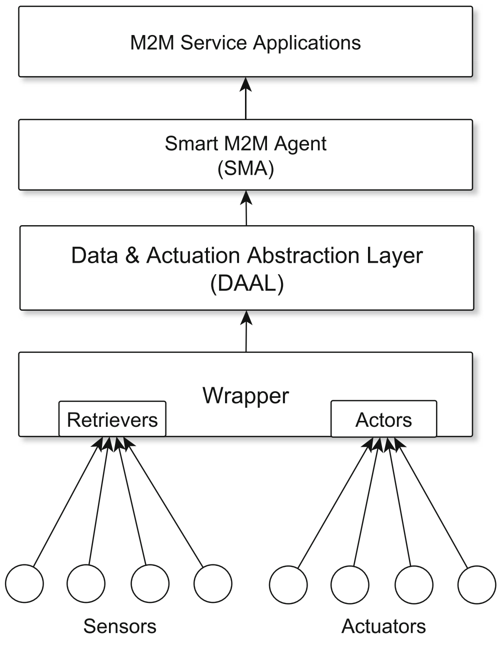


Fig. 10 A high-level ISIS architecture

previously mentioned issues, i.e., identification, resource management and partially semantic interworking. In order to address some challenges mentioned in Section 1, ISIS takes the generated resource data model from SENSEI as a basis for its entity model and incorporates some standard technologies such as device abstraction from ETSI M2M and OMA DM.

Figure 10 shows the high-level system architecture of ISIS. The key of ISIS is the Smart M2M Agent (SMA) running on each of the machines such as M2M device, M2M gateway and M2M server. Each SMA reads the sensors available on its device and shares the data with all other interconnected SMAs. For interoperability and portability, ISIS includes two layers, Data and Actuation Abstraction Layer (DAAL) and Wrappers, between SMA and its physical sensors.

Identification and discovery Similar to the SENSEI project, ISIS defines an entity/attribute based data model as shown in Fig. 11. Each information produced by M2M devices is described as an entity that consists of one or multiple context attributes such as a name, unique identifier, type, etc. When entities are registered at a SMA, they are stored in a local database and made available in the network. A query/subscription is sent to the SMA, and then the SMA looks for its internal database and returns all matching entities. With these attributes, all M2M devices and information available within ISIS can easily be identified and discovered.

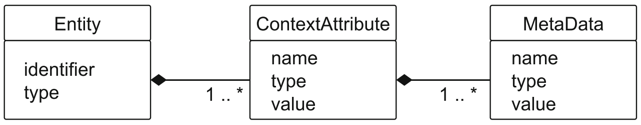


Fig. 11 ISIS entity/attribute based data model

Resource management For resource management, ISIS directly exposes OSGi compatible API for M2M applications and supports additional interface proxies, i.e., XML-RPC over HTTP [11] and NSGI-10 [1]. Since M2M applications managing M2M devices can be implemented in using different languages, language neutrality is an important issue that needs to be considered from the start therefore the ISIS platform supports the implementation of XML-RPC on top of the SMA. ISIS also supports a REST-binding for OMA NGSI APIs to provide convenience device management functions following a web-of-things approach. This enables a management client to use a simple URL-based access to all entities and their attributes in the network.

Interoperability and semantic interworking A main component of the ISIS system is Data&Actuation Abstraction Layer (DAAL). DAAL is used to support the exchange of information between physical M2M devices and Agent. DAAL bridges attached physical M2M devices using different underlying network technologies (e.g., Zigbee) into a SMA through mapping raw data into attribute values of the entity that the SMA is running. DAAL implements all common functionalities that are used to discover/manage different types of M2M devices. In addition, DAAL is an intermediary in dispatching the requests form the SMA and notification updates generated by asynchronous sources.

Evaluation The main goal of our evaluation is to test the feasibility of ISIS. We measure ISIS based on the following three parameters :

- *Stability* was measured by the error free execution time of the platform in a given setup.
- *Scalability* was measured by the number of handled devices for which the stability of execution is assured.
- *Robustness* was quantified as a measure of stability of the platform in presence of failures.

ISIS is implemented in Java. For the test, we used a 2.53 GHz Intel(R) Core(TM)2 Duo machine with 3.9 GB of RAM. ISIS ran on Java Virtual Machine 1.7 with 512 MB memory.

We performed functionality and performance tests to check the correctness of the functions and the platform efficiency, respectively. For the functionality test, we ran ISIS with set of Z-Wave[16] based sensors measuring power level during a 103 day time period. During the period, ISIS

[16]http://www.z-wavealliance.org/

correctly handles 1,483 milion sensor readings and properly operates for various events such as detecting a new sensor attachement and checking the disconnection of sensors.

The performance tests were used to qualitatively measure the performance of the ISIS platform regarding the number of sensors to handle and the frequency with which they send the data. For this test, we configured a sensor to keep sending message every certain period of time (i.e., between 100 to 1000 miliseconds) according to normal distribution for a given test duration 60 seconds and varied the number of sensors from 1 to 100. Each test case is executed until ISIS receives the expected number of messages. The expected number (N) is calculated by using:

$$N = S \times 60000ms/T$$

where S is the number of sensors and T is the period in milliseconds.

The test is stopped once this number of message is achieved. Afterwards, we measured the overhead time (%), i.e., how many more time was required than for the delay free execution, which means that ISIS receives the expected number of messages within 60 seconds.

Figure 12 shows the results of the tests. Considering a typical number of sensors (12–20) used for HVAC (Heating, ventilation and cooling) [12], the results indicate that the performance of ISIS is stable enough with reasonable overhead time even when it uses high number of sensor (100) and a common used event period (1 second).

Another performance tests were performed in order to study the relationships between memory usage and maximum number of devices. We also intend to analyze the viability of deploying the prototype in a real-world environment.

For the tests, we varied the number of devices from 52 to 1024, which are the number of devices of a home and a medium-sized hotel. After configuring one of pre-defined number devices, we sent different number of messages per minute (i.e., 0.16 to 2727) and checked the failure of the system (i.e., memory error). If the system runs more than one week without any memory error, we marked the test run as success. We also limited the JVM heap memory size to 64 MB so that we could check actual configuration conditions that can be used to run ISIS.

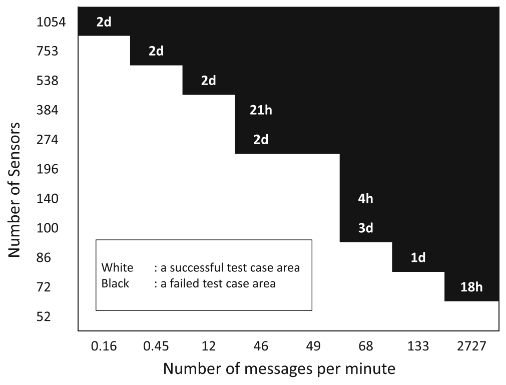


Fig. 13 Error occurrence

Figure 13 shows the results of the tests. Overall, this Figure could provide a guidance to developers or service providers who use ISIS how to deploy ISIS with their specific operating enviornment. As the Figure shows, ISIS is able to run with 52 sensors with 2727 messages per minute (22 milliseconds period) without any errors for a week. Another finding is that With 1054 sensors and 0.16 messages per minute, ISIS reached out of memory state with the current setup after 2 days running. It is useful to support another test scenario, i.e., increasing heap memory size. Meanwhile, it is stable with 753 sensors.

In order to check the impact of the operating memory on ths stability and scalability of the ISIS platform, we performed another experiment. In this scenario we focus on heap, where JVM stores all objects created by Java application, as an operating memory factor. Our test cases use various numbers of sensors, and these numbers consequently impact the number of stored objects.

Previously we used 512MB heap memory size while 10 % additional memory is used in this scenario. Other than

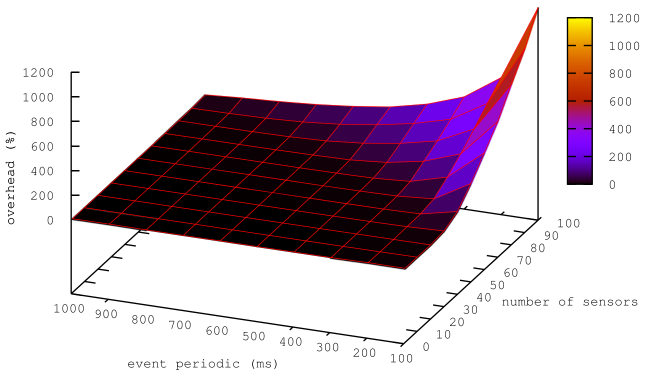


Fig. 12 Overhead time impact

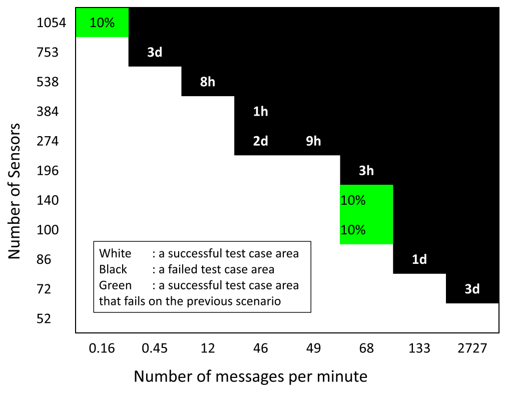


Fig. 14 Impact of operating memory

that the same environment was applied here. As Fig. 14 shows by 10 % additional memory, the test with 1054 sensors and 0.16 messages per minute is able to run without error in a week (see green colored cell). The scalability factor appears here. At some points, increasing number of sensor only works if we also increase number of memory.

Additionally, we also performed tests in a constrained environment which uses very low JVM memory namely 64MB. The 64MB case was tested for less cases and results show that it works for up to 1.2 messages per minute for 274 devices and it fails for 0.45 messages with 384 devices.

From these performance experiments we can see that ISIS can reliably support small to med-size home or building environment running on JVM with low memory (64MB). For a large size environment, as a simple solution, developer could use larger JVM heap memory size. However, a fundamental solution would be to use distributed SMA to scale up the number of managing devices. We plan to continue to enhance the ISIS platform in terms of scalability so that ISIS can support large size operational environment such as a large size hotel.

5 Summary and conclusions

This article presented how M2M devices in the future network can be efficiently connected and managed in M2M systems. Although M2M architectures including the ones from standard organizations such as ETSI TC M2M and 3GPP provide a set of functionalities for managing and controlling M2M devices, they only provide a minimal deployable solution at the moment and are limited in the aspect of resource management, overload control, device discovery, semantic interoperability and conflict management. The focus of this article was mainly on investigating solutions to address these issues. We introduced the current available solutions for these challenges followed by an analysis to highlight required enhancements.

Furthermore, this paper presented the realizations of the new M2M framework for managing and connecting M2M devices through two EU projects, SENSEI and CAMPUS 21. The SENSEI project provides a fundamental basis for modeling the real world resources. The proposed resource model introduced a novel way of presenting information semantics from low sensor data to enrich sensor data that can be used by multiple M2M Applications without prior knowledge. This also enables to add several advanced functionalities such as semantic device discovery and conflict resolution. To enable further extend the architecture and its resource model to support the management and control of a large number of heterogeneous devices, the CAMPUS 21 project has been started.

In CAMPUS 21, we are responsible for developing the core M2M middleware platform. We introduced a prototype implementation of the ISIS platform showed the feasibility and proof of concept of CAMPUS 21. Our evaluation shows that the current version of ISIS can reliably run in very low memory of JVM (64 MB) with a lot of different types of sensors and high-speed frequency although still requires further enhancement in terms of scalability.

We are now extending the proposed implementation to handle other addressed shortcomings such as semantic support and conflict resolution. We believe that developing the CAMPUS 21 architecture focusing on management and control mechanisms based on the resource model enhanced from SENSEI will provide a fundamental basis for future M2M frameworks.

References

1. OMA Next Generation Services Interface Requirements (2012) OMA-RD-NGSI-V1-0, http://www.openmobilealliance.org/
2. Bandyopadhyay D, Sen J (2011) Internet of things: applications and challenges in technology and standardization. Springer Wirel Pers Commun 58(1):49–69
3. CAMPUS 21: Deliverable 4.2, specification of systems architecture and concept for integration and upscaling: integration methods systems architecture, and middleware (2012) http://www.campus21-project.eu
4. Chang K, Soong A, Tseng M, Xiang Z (2011) Global wireless machine-to-machine standardization. Internet Comput IEEE 15(2):64–69
5. Charles P, Grothoff C, Saraswat V, Donawa C, Kielstra A, Ebcioglu K, von Praun C, Sarkar V (2005) X10: an object-oriented approach to non-uniform cluster computing. SIGPLAN Not 40(10):519–538
6. Cheng J, Law KH (2002) Using process specification language for project information exchange. In: 3rd international conference on concurrent engineering in construction, pp 63–74
7. Dunlop N, Indulska J, Raymond K (2002) Dynamic conflict detection in policy-based management systems. In: Proceedings of the sixth international enterprise distributed object computing conference (EDOC'02), EDOC '02. IEEE Computer Society, pp 15
8. Dunlop N, Indulska J, Raymond K (2003) Methods for conflict resolution in policy-based management systems. In: Proceedings of the 7th international conference on enterprise distributed object computing, EDOC '03. IEEE Computer Society, pp 98
9. Jeronimo M, Weast J (2003) UPnP design by example: a software developer's guide to universal plug and play. Intel Press
10. Kunz A, Kim L, Kim H, Husain S (2012) Machine type communications in 3GPP: from release 10 to release 12. In: GLOBECOM workshops (GC Wkshps), 2012 IEEE
11. Laurent SS, Dumbill E, Johnston J (2001) Programming web services with XML-RPC. O'Reilly & Associates, Inc, Sebastopol, CA
12. Lu J, Sookoor T, Srinivasan V, Gao G, Holben B, Stankovic J, Field E, Whitehouse K (2010) The smart thermostat: using occupancy sensors to save energy in homes. In: Proceedings of the 8th ACM conference on embedded networked sensor systems, SenSys'10. ACM, pp 211–224

13. Ostendorf K (2010) Description and specification of open M2M API. Open API for M2M applications
14. Pandey S, Kim MS, Choi MJ, Hong J (2011) Towards management of machine to machine networks. In: network operations and management symposium (APNOMS), 2011 13th Asia-Pacific, pp 1–7
15. Pautasso C, Zimmermann O, Leymann F (2008) Restful web services vs. “big” web services: making the right architectural decision. In: Proceedings of the 17th international conference on world wide web, WWW ’08. ACM, New York, pp 805–814
16. Presser M, Barnaghi P, Eurich M, Villalonga C (2009) The sensei project: integrating the physical world with the digital world of the network of the future. IEEE Commun Mag 47(4):1–4
17. Steinberg DH, Cheshire S (2005) Zero configuration networking: the definitive guide
18. Taleb T, Kunz A (2012) Machine type communications in 3gpp networks: potential, challenges, and solutions. IEEE Commun Mag 50(3):178–184
19. BBF TR-069, CPE WAN Management protocol, issue 1, Amendment 4 (2011) http://www.broadband-forum.org/technical/download/TR-069_Amendment-4.pdf
20. 3GPP TR 23.887, Architectural enhancements for machine type and other mobile data applications communications (2012) http://www.3gpp.org/ftp/Specs/html-info/23887.htm
21. 3GPP TS 23.060, General packet radio service (GPRS); service description; Stage 2 (2012) http://www.3gpp.org/ftp/Specs/html-info/23060.htm
22. 3GPP TS 23.204, Support of short message service (SMS) over generic 3GPP internet protocol (IP) access; stage 2 (2012) http://www.3gpp.org/ftp/Specs/html-info/23204.htm
23. 3GPP TS 23.401, General packet radio service (GPRS) enhancements for evolved universal terrestrial radio access network (E-UTRAN) access (2012) http://www.3gpp.org/ftp/Specs/html-info/23401.htm
24. 3GPP TS 23.682, Architecture enhancements to facilitate communications with packet data networks and applications (2012) http://www.3gpp.org/ftp/Specs/html-info/23682.htm
25. Tsiatsis V, Gluhak A, Bauge T, Montagut F, Bernat J, Bauer M, Villalonga C, Barnaghi P, Krco S (2010) The SENSEI Real World Internet Architecture. In: Towards the future internet - emerging trends from European research, IOS Press, pp. 247–256