Information-Centric Networking for M2M Communications: Design and Deployment

Marica Amadeo, Orazio Briante, Claudia Campolo,
Antonella Molinaro, Giuseppe Ruggeri
University "Mediterranea" of Reggio Calabria - DIIES Department
Email: {name.surname}@unirc.it

March 19, 2021

Abstract

The European Telecommunications Standards Institute (ETSI) recently released a set of specifications for a reference architecture to globally access resources provided by machines over heterogeneous technologies in an interoperable way through a RESTful interface. Resources are named through Uniform Resource Identifiers (URIs) at the application layer and typically reachable at the network layer through IP connectivity. Such an approach can be used also to access *extremely resource-constrained* devices, provided that lightweight interactions with a gateway, remotely exposing their resources, are granted. Among potential alternatives to support communication between the gateway and such constrained devices, we investigate the Information-Centric Networking (ICN) paradigm, gaining momentum in the future Internet research arena. It differs from the host-centric IP networking in that it cares for the content to retrieve instead of the device hosting it. By directly using content *names* at the network layer and a receiver-driven communication, ICN well fits the requirements of many machine-to-machine (M2M) applications that are information-centric in nature and rely on a publish-subscribe service model.

In this paper, we propose an ICN-based solution to be deployed on top of constrained devices whose *named* resources are exposed at a wide area scope by an M2M gateway. The proposal aims at ensuring easy interoperability with ETSI M2M specifications, thus allowing remote applications to access the resources of ICN-enabled nodes. To showcase the viability of our proposal, a test-bed has been deployed leveraging low-cost devices for home automation. Experimental results confirm a good performance in terms of device resources consumption, easiness of implementation and latency of communication.

https://doi.org/10.1016/j.comcom.2016.03.009

1 Introduction

Machine-to-machine (M2M) communications aim at connecting devices that operate *autonomously*, i.e., without human intervention, and enable numerous services in smart contexts such as smart home, smart grid, smart transportation, among many others [1].

M2M systems are typically composed of devices equipped with sensors and actuators, organized in a M2M Area Network, whose resources can be accessed by local and remote applications (directly or through a gateway). Machines can be constrained with respect to energy, computation, storage, and bandwidth [2]. As a result, M2M Area Networks usually require low-power low-rate radio technologies (e.g., Bluetooth, IEEE 802.15.4) and ad-hoc defined lightweight networking and upper-layer protocols.

So far, different vertical M2M solutions have been designed separately for different application domains, thus hindering interoperability and large-scale deployment [1]. To facilitate the standardization process and limit the gap between M2M architectural proposals, the European Telecommunications Standards Institute (ETSI) released a set of specifications for a common M2M service platform. It is based on a RESTful (REpresentational State Transfer) approach with *open interfaces* that enable the deployment of services *independently of the underlying technologies* [3].

The ETSI M2M model is today embraced by many commercial platforms (e.g., ThingSpeak, Nimbits, EVERYTHNG [1]) and by the *oneM2M Initiative*, an international partnership project currently active to define a globally applicable, access-independent M2M *horizontal service layer* specification [4]. The OM2M project has been also proposed as an ETSI-compliant platform for M2M interoperability [5].

Since the Internet today provides IP-based connectivity, the tendency followed by ETSI M2M is to couple the RESTful paradigm, which addresses resources at the application layer via Universal Resource Identifiers (URIs), with the IP protocol, to ensure global access to services and information. The strengths of IP are universally well-known, however its flexibility is achieved through *extensions* conceived to cope with the demands of emerging M2M scenarios, e.g., resource-constrained devices require IP protocol adaptations like 6LoWPAN [6].

In parallel, the research community is also exploring alternative networking solutions for the future Internet. One of the most attractive proposal is Information-Centric Networking (ICN), whose benefits have been evaluated both in the core network [7] and in different wireless environments [8]. Lately, it showed high potential also in M2M systems [9]. In ICN, resources are identified by unique and persistent names, which are directly used by routers for search and provisioning operations in the network. This is a fundamental departure from the IP-based approach, where communication is host-centric, being based on the IP addresses of source and destination nodes.

ICN well matches the information-centric pattern of many M2M applications (e.g., measured data reports from meters in the smart grid, monitored temperature values in the smart home), which care about *what* data to retrieve (or service to request) instead of *which* node to connect to [9]. Through flexible name-based primitives, ICN would simplify M2M data/service access and facilitate the support of not only *single consumer/single source* interactions, but also group-based communications with *multiple consumers/multiple sources* interactions. The latter ones are typical in M2M systems, e.g., multiple stakeholders retrieving home energy measurements, multiple home sensors providing temperature values.

Recent works have evaluated the real applicability of ICN in local M2M scenarios, e.g., building automation [10], energy management in a smart home [11]. Large-scale deployment and evaluation of clean-slate M2M-ICN systems are however currently infeasible, since the global connectivity is IP-based. To allow remote applications to access resources of M2M-ICN devices, a convenient solution would deploy *interworking proxy capabilities* in special network nodes that connect the ICN and the IP domains, i.e., in the gateway connecting an M2M-ICN Area Network with the Internet.

In this paper, by elaborating on the concept of *interworking proxy*, we deploy a communication framework that enables interactions between remote consumer applications and the resources in an M2M-ICN Area Network. The approach exploits some similarities between the ICN name-based access to resources and the ETSI RESTful logic. In both cases, application-level identifiers (URI or names) are used to fetch the resources, but in ICN a name does not need to be mapped into the IP address of the node owning the resource. In the proposed functional architecture, the Gateway exposes the resources available in the M2M-ICN Area Network by following the ETSI M2M name structure, and the remote applications follow the well-defined procedures to access them [3]. Then the Gateway translates the incoming requests into ICN messages to be forwarded to the M2M devices to retrieve resources (or trigger actions) in the M2M Area Network. ICN names are designed according to the ETSI M2M name structure, thus facilitating the implementation of the proxy capability in the Gateway. At the same time, we customize the ICN primitives to support low-overhead lightweight communications in the M2M Area Network.

The viability of the proposed framework is assessed through an experimental test-bed leveraging extremely resource-poor devices in a smart home domain, as a representative M2M use case. Indeed, M2M communications may play a major role in houses, where automation of some processes appears essential in many home-related sectors such as comfort, health, security, energy efficiency, etc. Results demonstrate that ICN is an attractive solution in M2M contexts.

To the best of our knowledge, this paper is the first tentative of designing and providing evidence of an M2M system that supports communication on a global scale and benefits from both ICN communication and the ETSI M2M standardization efforts.

The remainder of the paper is organized as follows. Section 2 provides an overview of M2M communications based on the ETSI M2M architecture. Section 3 introduces the ICN paradigm along with a description of the main related literature. Section 4 describes our proposed M2M-ICN framework; Section 5

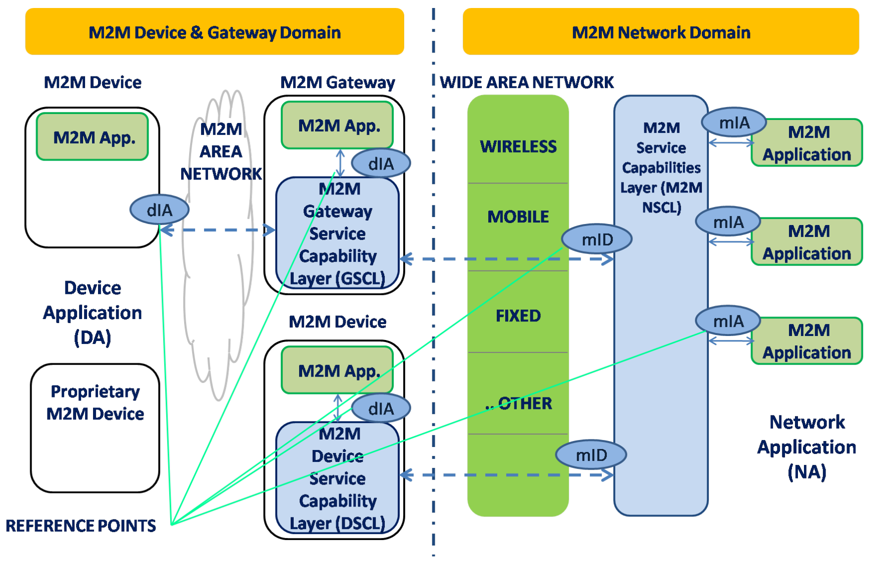


Figure 1: ETSI M2M Functional Architecture [3].

details the experimental test-bed that showcases the viability of the conceived framework. Performance evaluation is reported in Section 6. Finally, Section 7 concludes the paper.

2 ETSI M2M architecture

2.1 Basic model

The reference ETSI M2M functional architecture in [3] consists at high-level of two main parts, namely the *Device and Gateway Domain* and the *Network Domain*. As shown in Figure 1, in the Device and Gateway Domain, M2M devices can run M2M Device Application(s) (DAs) using M2M Service Capabilities. They can be directly connected to the Network Domain via the access network, or are connected to a Gateway through an M2M Area Network. The Gateway acts as a proxy between M2M Devices and the Network Domain and manages authentication, authorization, management, and provisioning procedures through its M2M Service Capabilities. The Network Domain includes the Access and Core Network together with M2M Service Capabilities, Network Applications (NAs) and a set of management functions.

The Service Capability Layer (SCL) includes the set of *common functions* to enable M2M communications between different entities. An SCL is therefore deployed on networks (NSCL), gateways (GSCL) and devices (DSCL), while reference points (*mIa*, *dIa*, *mId* in Figure 1) based on open Application Programming Interfaces (APIs) are specified that offer generic and extendible mechanisms for interactions between different SCLs and between Applications and SCLs [12]. A central role in the system is maintained by the NSCL, which is involved in mutual SCLs authentication, registration of NAs and D/GSCLs, resource discovery, subscriptions and provisioning operations.

2.2 Resource representation and management

ETSI M2M adopts a RESTful style to define how Applications and/or SCLs exchange information with each other. The REST paradigm implies a client/server model and is based on the notion of *resource*: anything in the system that can be named and addressed can be a resource, e.g., data, services.

ETSI M2M addresses all the resources residing in the SCLs by introducing a hierarchical tree which models the structure of the resources, the relationships between them and their properties, such as registered SCLs, registered applications, access rights, subscriptions. The same tree structure applies to resources in NSCL, GSCL and DSCL. As shown in Figure 2, the root for all resources on a hosting SCL is called <sclBase> and includes a set of child resources, e.g., a collection of <scl> resources representing remote SCLs with which the hosting SCL is registered to.

As a result, resources are uniquely addressable and identifiable via URIs, which are obtained by concatenating the name components in the hierarchical tree. Resources are transferred and manipulated by the CRUD (*Create*, *Retrieve*, *Update*, and *Delete*) verbs over different transport protocols, e.g., Hyper Text Transfer Protocol (HTTP). Two additional verbs are also considered in M2M: *(i) Notify*, to report a resource change as a consequence of a subscription, *(ii) Execute*, to execute a management command/task.

In general, the access to M2M resources from consumer applications can be handled in two main ways: *request-response* and *publish-subscribe*. The request-response scheme enables synchronous interactions: the issuer sends a request and receives the information in a response message. In the publish-subscribe scheme, the issuer subscribes to a resource and is notified of the resource status periodically or when it changes. Thanks to the definition of the <subscription> resource in the hierarchical tree, publish-subscribe mechanisms are standardized by the ETSI M2M architecture, which denotes an active subscription to the resource identified by a *resourceURI* with the name: *<resourceURI>/subscriptions/<subscription>* and includes several attributes like *expirationTime* and *minimalTimeBetweenNotifications*.

2.3 The case of constrained devices

By standardizing the structure of resources that reside on SCLs and the procedures for handling them, ETSI M2M enables the development of M2M services with a global scope and in presence of heterogeneous devices. Resource-constrained nodes unable to implement the SCL are reachable through the Gateway, which exposes all the available resources to the NSCL.

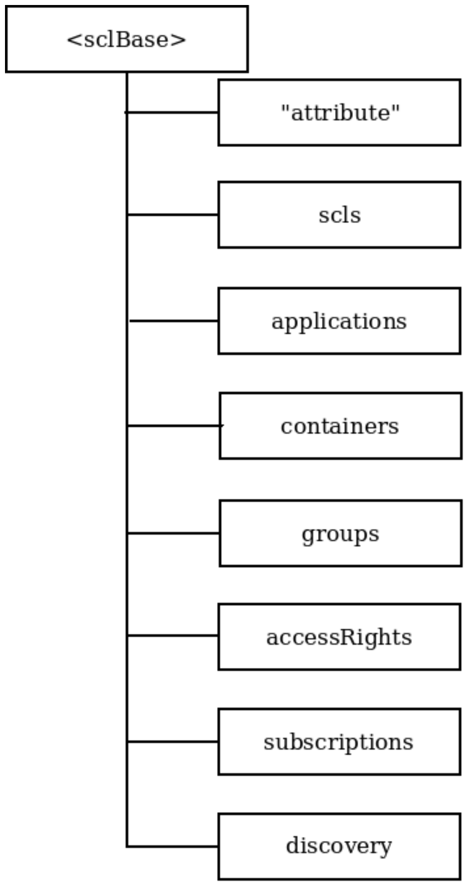


Figure 2: Structure of <sclBase> resource tree [3].

Thanks to the RESTful approach, a binding between the defined M2M REST resources/primitives and the HTTP REST resources/methods can be easily done [12]. Similarly, a mapping can be performed with the Constrained Application Protocol (CoAP), which has been standardized to introduce the web service paradigm into low power and lossy networks with constrained devices [13].

A rough taxonomy of constrained devices, according to their device capabilities, can be found in [14]: *class 0*, *class 1*, *class 2*, in increasing complexity order. *Class 0* devices due to memory or programming limitations (RAM $\ll$ 10 KiB, Flash $\ll$ 100 KiB) will participate in Internet communications with the help of larger devices acting as proxies, gateways, or servers. *Class 1* devices are quite constrained in processing capabilities, they cannot easily talk to other Internet nodes employing standard protocol stack (based on HTTP) and use CoAP over the User Datagram Protocol (UDP). *Class 2* devices are basically capable of supporting most of the same protocol stacks as used on laptops and servers.

ETSI M2M also considers non-compliant devices (e.g., nodes without REST/IP connectivity) and allows them to connect by using an interworking proxy capability, which can be located in the SCL of an interfacing node, e.g., the Gateway in Figure 3. However, interworking proxy capabilities are out of the scope of the standardization document [3].

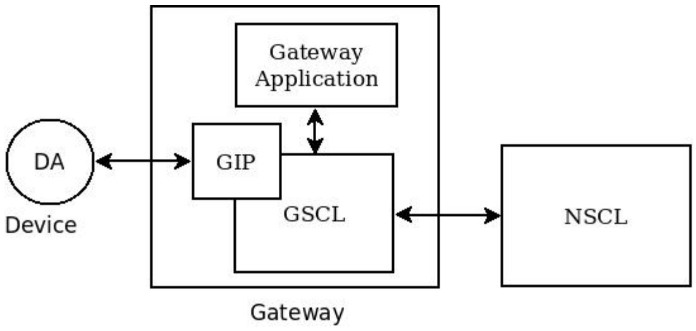


Figure 3: Deployment scenario with Gateway Interworking Proxy (GIP) Capability.

In this paper, we focus on extremely constrained devices (i.e., *class 0*) and implement for them a lightweight protocol stack based on ICN. The Gateway runs the GIP functionality so ICN devices can participate to the ETSI M2M system. One of the target of this work is indeed the GIP implementation to enable interactions between ICN and ETSI M2M systems and show the ICN potential within a global standardized architecture.

3 ICN for M2M communications

ICN in a nutshell. ICN is a new communication paradigm centered around content names: an ICN consumer application requests a content by directly using the content name, and the request is routed in the network until the content is found. The content packet transmitted by the producer carries a correspondent unique and persistent name and some information for security purposes. As a result, *in-network caching* is extremely facilitated, by letting potentially every network element cache the data packets that traverse it, so that they will be available to serve future requests. By doing so, *asynchronous* communication is enabled that effectively decouples consumers and producers.

Related work. Originally proposed for content dissemination in the Internet, today ICN is an appealing solution also in other domains, including Internet of Things (IoT) [15, 16, 17] and M2M [9]. In fact, by enabling direct name-based routing and in-network caching, ICN *(i)* does not need resolution systems that translate application-level names into IP addresses, *(ii)* simplifies and speeds up resource discovery and delivery, and *(iii)* well-matches information-centric applications.

So far, a few recent works considered ICN in local M2M scenarios, e.g., to enable building automation systems [10, 18, 19] and smart energy management [11]. Specifically, in [10], the initial design of a secure Building Management System is presented, with focus on a data sensor acquisition system that implements encryption-based access control. The system uses a proprietary protocol for the communication between sensors and the gateway that collects the sensing data. The gateway publishes data in ICN repositories and responds to requests of authorized users.

In [18], the initial design of an ICN based homenet is presented with a focus on naming and service configuration. The ICN logic is only in the home gateway and interior routers, i.e., the *powerful* nodes, but not in the sensors. A comparison against an IPv6-based home network shows that ICN significantly reduces the traffic overhead. The case of securing a lighting control system running over ICN is discussed in [19]. The proposed framework includes a *configuration manager*, which assigns fixtures and applications their namespace and identity, represented by a unique public/private key pair; and an *authorization manager*, which determines the applications allowed to access each fixture, signs applications' public keys and issues signed access control lists.

An ICN-based system for home energy management services called iHEMS is presented in [11]. The work defines a set of secure publish/subscribe primitives built on top of ICN to exchange environment sensing and power data.

The potentialities of ICN in a large scale M2M scenario is discussed in [9], where a smart grid domain is considered. A high-level architecture is proposed where ICN forms a publish/subscribe overlay network that includes end-systems (e.g., phasor measurement units, control centers) and special network nodes which act as rendezvous point and organize information in location-independent topics. A feasibility study simulating a real power distribution network in the Netherlands demonstrates the ability of ICN to disseminate topic-based information with low latency.

A first work that introduces ICN in the ETSI M2M architecture is in [20], where the authors conceive the idea of an ICN-based Overlay Service Capability Layer that improves the NSCL functions by realizing distributed service discovery, peer-to-peer subscription, data exchange and Quality of Service (QoS) monitoring.

Our work differs from [20] since we do not alter the ETSI M2M architecture, but we define a new GIP capability to allow interoperability with an ICN M2M Area Network. The resulting framework allows remote NAs registered to the NSCL to interact with constrained devices through light ICN primitives following a *demand-response* or a *publish/subscribe* model.

4 Extending the ETSI M2M architecture with ICN principles

4.1 Basics

In our study we consider an M2M Area Network, where resource-constrained devices, equipped with a low-power wireless access technology, e.g., IEEE 802.15.4, expose their services through an M2M Gateway. We rely on ICN as a networking technology between the devices and the Gateway, which also implements the GIP functionality and the GSCL to communicate with the ETSI M2M Network Domain.

Among several information-centric architectures [7], we refer to *Named Data Networking* (NDN) [21] for the design of the M2M-ICN Layer. Indeed, NDN offers important features that facilitate interworking with ETSI M2M systems.

First, NDN uses hierarchical, sometimes user-friendly, namespaces to name resources. NDN names are URI-like strings with a variable number of components and virtually unbounded lengths. This implies that NDN names can be built that match the ETSI M2M resource tree structure described in Section 2.2.

Second, NDN facilitates wireless networking because communication is just based on two packet types, *Interest* and *Data*, used respectively to request by name the content and transfer it. Interests are usually broadcasted over the wireless medium [22], thus any node holding the Data can answer the request.

Third, NDN natively supports *multi-consumer* communication through Interest aggregation and Data caching. Specifically, each node maintains three tables: the Content Store (CS), used for caching incoming Data, the Forwarding Information Base (FIB), used to route Interests, and the Pending Interest Table (PIT), used to track the forwarded (and not yet satisfied) Interests and the arrival interface(s), thus Data can be sent back to the requester(s). The M2M Gateway, which bridges the local network to the Internet, can receive queries or subscriptions from many external consumer applications interested in the same data (e.g., the house owner and the smart grid stakeholders interested in energy consumption data). Interests with the same name can be aggregated in the PIT of the Gateway and transmitted only once, thus limiting the number of accesses to the device and saving network and energy resources. Moreover, requests can be also satisfied by using the Data cached in the CS of the Gateway, if not stale, so to reduce the interaction with constrained devices.

Multi-source communications (i.e., to target more resources simultaneously with a single Interest, instead of transmitting multiple separate requests) can be also handled by NDN, provided that name conventions and packet processing rules are defined [23].

In the following, we present our communication framework by describing *(i)* the design of the NDN naming scheme for M2M communications, *(ii)* the design of NDN multi-source communications, *(iii)* the interactions between remote NAs and NDN constrained-devices across the ETSI M2M systems.

4.2 Naming scheme

According to the ETSI M2M specifications, each DA, running in devices able to host an SCL, is modeled as a resource, identified by a unique name and associated with a set of hierarchically organized sub-resources, e.g., each content produced by the application can be identified by the string: */applications/-<app>/containers/<container>/contentInstances/<instance>*.

In our proposal, NDN names could be defined according to the ETSI M2M hierarchical tree and directly used to fetch data by leveraging the Interest/Data exchange. For instance, in a smart home, the Gateway could ask for the current value of the temperature in the kitchen by sending an Interest with name: */applications/TemperatureSensing/containers/kitchen/contentInstances/current-Value*.

However, it is worth noticing that some M2M access technologies (e.g., IEEE 802.15.4) support small payloads and names should be maintained thin to avoid packet fragmentation and limit the load on constrained devices.

To this aim, we define an NDN namespace characterized by short, abbreviated names that can be used locally and, on the occurrence, easily mapped into ETSI M2M names to enable interactions with the external Network Domain.

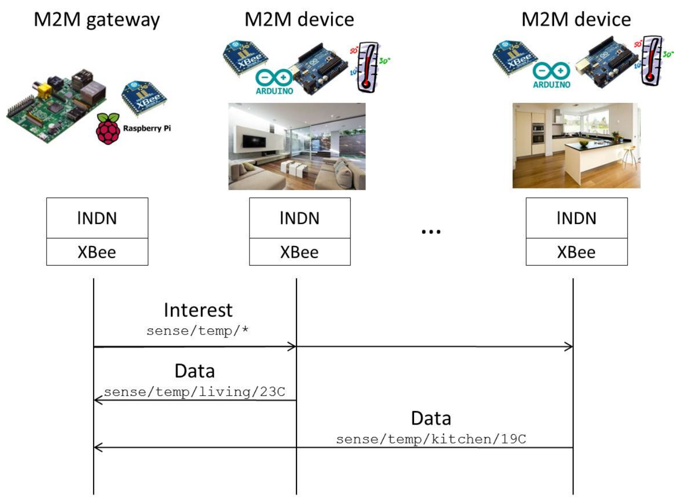
The proposed naming scheme is application-specific and currently designed to support simple services offered by sensors and actuators in a local environment. Basically, each sensor in our scenario is provided with a *sensing DA* (sDA), whose only task is to generate a single type of measurement in response to a query (request-response pattern) or a subscription (publish-subscribe pattern).

An actuator, instead, executes a set of elementary actions, like *switch-on* and *switch-off* an appliance. The Gateway sends commands in Interest packets by using proper names that concatenate the identifier of the *actuation DA* (aDA) and the action type. Therefore, the namespace must codify not only the names of DAs, but also every resource related to them.

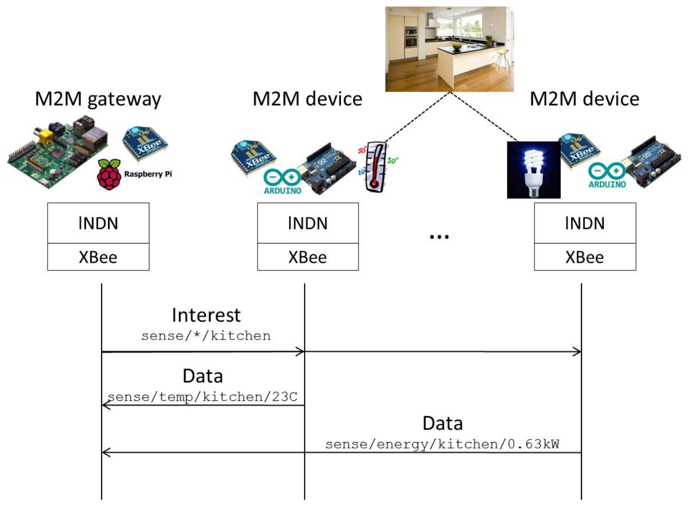
As regards the namespace setting, we assume that the DA name is composed of two parts: *(i)* a two-component principal prefix that describes the general application task, e.g., *sense/temp*, *action/airConditioning*, *(ii)* a set of subsequent components that further qualify the application, e.g., the location. The principal prefix can be shared by different devices, e.g., there are many temperature sensors in a house, but the concatenation of all the name components uniquely identifies a DA in the local environment. For instance, in a home network, the sDA that reports the temperature in the kitchen can be named as *sense/temp/kitchen*, while the sDA that reports the temperature in the bedroom can be named as *sense/temp/bedroom*.

Each DA-related resource is identified by the DA name followed by other components that characterize the resource, e.g., the temperature in the kitchen in a specific time instant can be identified by the name *sense/temp/kitchen/data/-<timestamp>*, the command for switching-off the air conditioning in the kitchen can be defined as *action/airConditioning/kitchen/command/<off>*.

The namespace is assigned to devices during the registration procedure with the Gateway.



(a)



(b)

Figure 4: Interest/Data exchange with the wildcard option set to query devices according to the offered resources (a) and to their physical location (b).

4.3 Multi-source communications

In addition to the perfect name matching that allows to query a specific DA, we define name conventions that enable a Gateway-initiated procedure for multi-source communications in the M2M Area Network. This would allow the Gateway to query more devices with a single Interest, with consequent advantages in terms of reduced traffic and overhead.

To this purpose, we introduce in the designed namespace the *wildcard* character "$*$" as part of the resource name to be carried in the multi-source Interest. The "$*$" character is used as a name component to indicate "*all resources that share the parts of the name preceding and/or following the* $*$ *character*".

Without loss of generality, DAs can be grouped *(i)* logically based on offered resources, or *(ii)* physically based on their location.

For instance, the common need to retrieve many sensing parameters at once (e.g., temperature, humidity, energy consumption) in a given location, e.g., a room in a house, could be supported by sending an Interest packet with name e.g., *sense/*/kitchen* to obtain the data from all sensors in the kitchen. The first name component, *sense*, in fact, identifies the first part of the application's principal prefix; the second name component would identify the sensing data type (e.g., temperature, humidity), but in this case the wildcard character is used to indicate *any type* of sensing; finally, the third component identifies the location, i.e., the kitchen.

If the Gateway intends to retrieve the temperature data from all the rooms in the house to decide if switching-on the heating system, it will send only one Interest with the name */sense/temp/*/.* Such name identifies all DAs that use the prefix */sense/temp/*, wherever they are located. The third component of the name would represent a location parameter, e.g., kitchen, bedroom, but the wildcard character is used in this case to indicate *any location.* Therefore, when receiving the Interest, all temperature sensors deployed in the house will answer to the request.

It is worth noticing that also in the vanilla NDN, thanks to the longest-prefix matching rule, all sensors sharing the name prefix */sense/temp* could reply with a Data packet upon receiving an Interest with name */sense/temp.* However, the *1-to-1 Interest-Data matching* of vanilla NDN does not allow to collect data from multiple sources with the one Interest. In fact, with NDN a pending entry is deleted from the PIT upon the first Data packet arrival, and successive Data packets matching the same Interest are consequently discarded. Here, instead, we disable the PIT entry deletion upon receiving a Data packet as a reply to a multi-source Interest. The deletion is associated to a timeout, set large enough to accommodate the replies from the queried devices, whose number is known in advance thanks to a preliminary registration procedure.

Examples for the two types of Interest/Data exchanges are shown in Figure 4.

4.4 Resource access via ETSI M2M

Services offered by DAs in the M2M Area Network can be accessed by an external NA provided that the Gateway is registered to the NSCL and maintains the resources representation in the GSCL. The GIP functionality in the Gateway translates NDN names into ETSI M2M names (and vice-versa), while the

GSCL follows the standard ETSI hierarchical tree representation to expose the resources. Thanks to the designed namespace, translation rules, implemented by the GIP, are quite easy, as discussed in the following.

As shown in Figure 5, we assume that the principal prefix of the DA name is used as application identifier in the GSCL, while subsequent name components are mapped as sub-resources, e.g., containers, subscriptions, contentInstances. This implies, for instance, that all the temperature sensing applications in a smart home are placed under the *senseTemp* application resource name, while the *location* name component is treated as a container sub-resource. Hence, if an NA wants to be notified about the temperature values in a room, it can subscribe to a content instance of the *senseTemp* application, by following the standard ETSI M2M routines. Then, the Gateway translates the subscription into an Interest packet and enables the indirect communication between the temperature sensors and the NA.

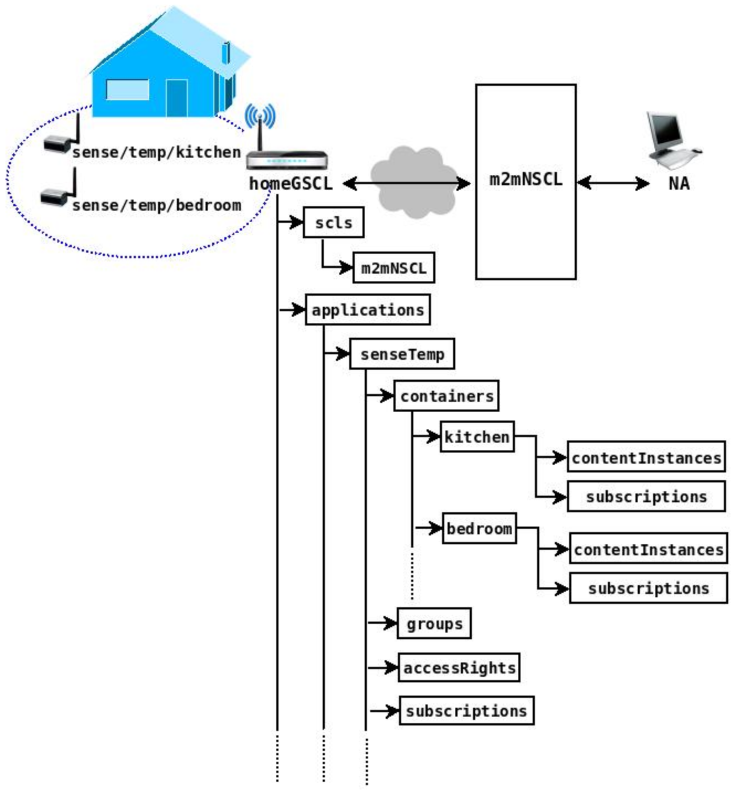


Figure 5: Translation of NDN names in the ETSI M2M format in a smart home.

In our design, we use long-lived Interest[1] as subscription packets: when an

[1]The term *long-lived* means that the Interest has a longer validity time w.r.t. the legacy

NA wants to subscribe to a resource, the Gateway sends an Interest with a name composed of the resource name and a set of subscription attributes defined by ETSI M2M: */resourceName/subscription/subscriptionAttribute1/.../-subscriptionAttributeN*. Currently, as mandatory attributes of the subscription, we consider the *exprTime*, which is the validity time of the subscription, and the *typology*, which can be *periodic* or *persistent*. A *periodic* subscription enables the device to periodically send the data (the time period between subsequent measurements is specified in the Interest), while a *persistent* subscription enables the device to send the data whenever a modification of the measured parameter (or event) occurs.

As an example, the subscription to obtain temperature data from the sensor in the kitchen at every 5 minutes for 60 minutes will be: */sense/temp/kitchen-/sub/exprTime/60/periodic/5*. A subscription Interest remains active in the PIT of the Gateway until the expiration time, thus allowing the reception of multiple Data from the producer DA. Then it can be renovated or deleted.

The entire procedure is shown in Figure 6, which describes the subscription to a content reporting temperature values in the kitchen.

The simpler case of synchronous request-response communications can be of course implemented, where the single query from a NA is converted in a legacy Interest packet, without the subscription attributes.

The case of multi-source communications can be also easily implemented. In fact, ETSI M2M defines the so-called *group* resource to identify and access groups of resources with a single query. Thanks to the proposed names conventions, a group-based query from a NA can be translated into a multi-source Interest in the M2M Area Network.

5 The deployed Test-bed

To showcase the viability of the proposed framework we built a demonstrator that implements a smart home system, as the one depicted in Figure 7, with off-the-shelf low cost devices. A remote NA retrieves information about the home environment by leveraging the ETSI M2M framework augmented with ICN functionalities in the M2M Area Network.

Details about the hardware and software modules are provided in the following.

Devices features. The Gateway is implemented over a Raspberry Pi device [24], which is a single-board computer equipped with a SD memory card, an Ethernet interface and a XBee [25] external interface, which is IEEE 802.15.4e compliant [26]. M2M devices are temperature and humidity sensors attached to open-source Arduino micro-controller boards [27] (as representatives of *class 0* devices) including XBee shields for communications with the Raspberry Pi. The choice to use Arduino for prototyping is because it is a highly flexible and cost-effective solution, providing a processor core, memory and input/output

Interest that retrieves only a single Data packet. The validity time corresponds to the subscription time.

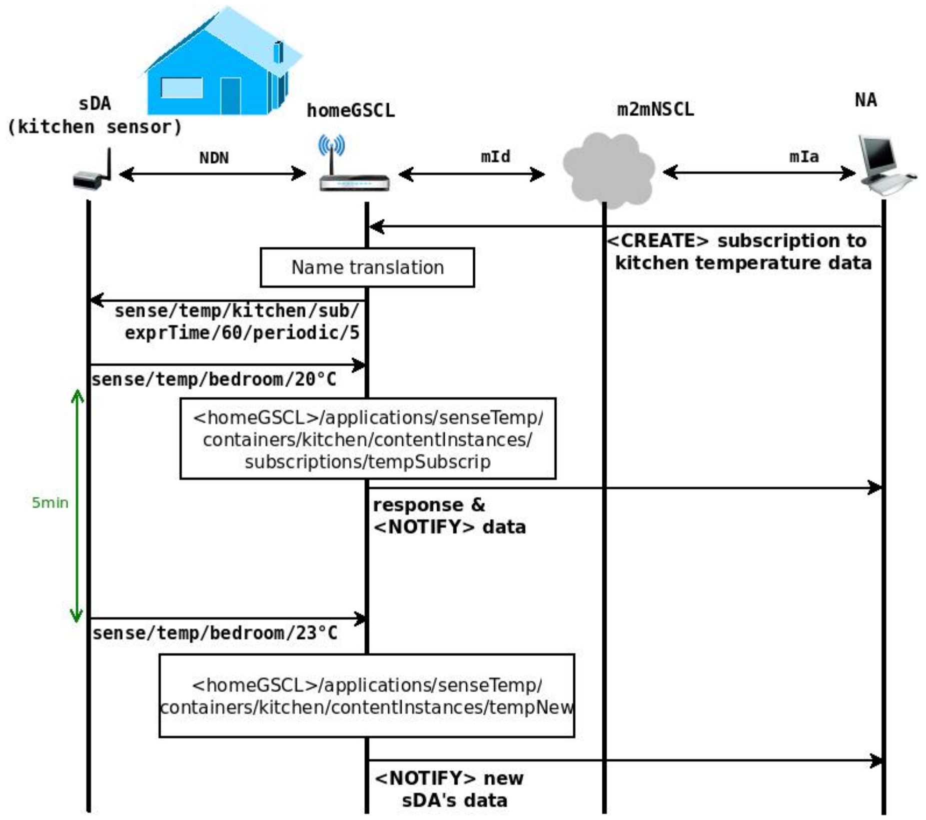

sDA
(kitchen sensor)
homeGSCL
m2mNSCL
NA
NDN
mId
mIa
<CREATE> subscription to kitchen temperature data
Name translation
sense/temp/kitchen/sub/ exprTime/60/periodic/5
sense/temp/bedroom/20°C
5min
<homeGSCL>/applications/senseTemp/ containers/kitchen/contentInstances/ subscriptions/tempSubscrip
response & <NOTIFY> data
sense/temp/bedroom/23°C
<homeGSCL>/applications/senseTemp/ containers/kitchen/contentInstances/tempNew
<NOTIFY> new sDA's data

Figure 6: Subscription procedure in a smart home: a NA subscribes to obtain the temperature in the kitchen.

peripherals, that can be attached to any variety of sensors. Each M2M device is one-hop far away from the Gateway. The NA runs over a remote Desktop-PC, which also hosts a web interface to explore and monitor the M2M resources. Finally, a workstation is used to host the NSCL.

Table 1 details the hardware features of M2M devices and the Gateway.

M2M-NDN implementation. The OM2M open source implementation of the ETSI M2M standard[2] is used to deploy GSCL and NSCL modules, running over the Gateway and the workstation, respectively. Such modules are not modified, but an interworking function (i.e., the GIP) is implemented in the Gateway to enable the communication between the GSCL and the NDN network, as it will be clarified in the following.

In the M2M Home Area Network, the M2M devices and the Gateway run a customized *lightweight NDN* (*lNDN*) implementation, (called *lNDN-arduino*

[2]OM2M project available at http://eclipse.org/proposals/technology.om2m/.

Table 1: Hardware features of M2M Gateway (Raspberry Pi) and M2M devices (Arduino)

Feature	**Raspberry Pi**	**Arduino**
CPU	ARM1176JZF-S	ATmega 2560
Clock speed	700 MHz	16 MHz
Storage type	Micro SD	Flash Memory
Storage	Max 32GB	256 KB
RAM	256 MB	8 KB

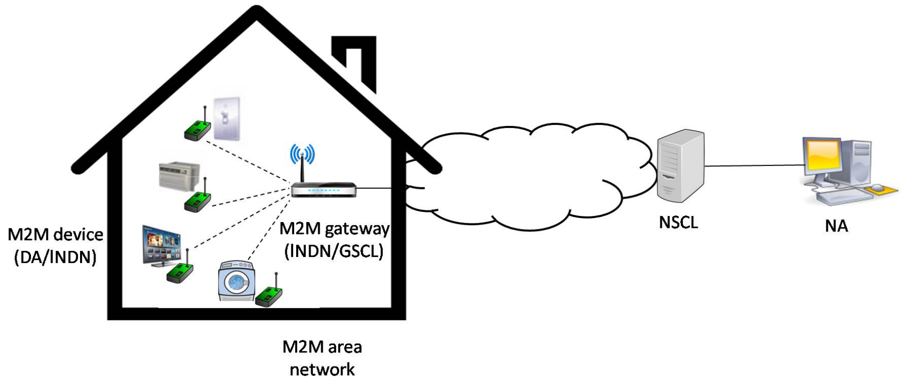


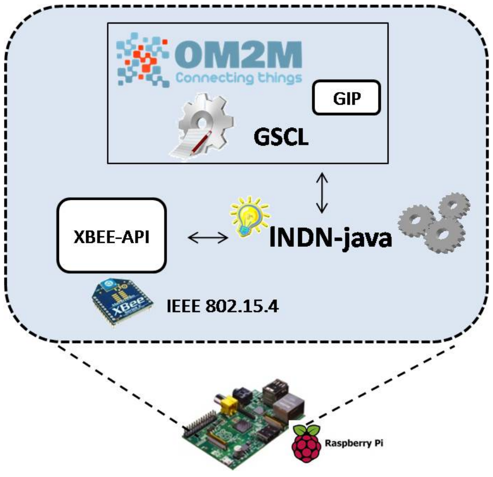
Figure 7: Reference scenario.

and *lNDN-java*, respectively), for in-home communications over IEEE 802.15.4. We originally devised *lNDN* in [28] to implement the basic NDN Interest and Data generation and processing; here it has been extended to support the M2M-ICN routines and features discussed in the previous Section, i.e., the DA namespace, group-based (multi-source) communications with the wildcard option, request-response and subscription-based delivery.

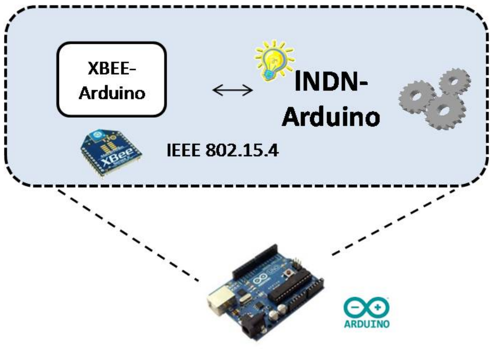
The implemented *lNDN-arduino* for the M2M device is a C/C++ library providing an abstract method that is customized depending on the sensor/actuator hardware characteristics. Vice versa, the *lNDN-java* library is written in Java and it can run over Linux-based systems.

The standard OM2M GSCL implementation at the Gateway is augmented with the newly defined GIP functionalities, targeting the interworking between the NDN-based M2M Home Area Network and the ETSI M2M system. The GIP module *(i)* creates a descriptor of the home resources in the GSCL by following the ETSI M2M URI format to make them accessible from remote NAs, and *(ii)* translates the ETSI M2M messages based on CRUD verbs into NDN packets and vice versa. For instance, when a NA requires the temperature in the kitchen with a *Retrieve* message carrying the ETSI M2M URI of the requested resource, the GIP translates it into an Interest packet targeting the kitchen temperature resource. When the named data is received, the GIP creates a response message containing the retrieved information and sends it back to the NA.

Figure 8 summarizes the software modules implemented in the Gateway and in the Arduino board. The *XBee-arduino* [29] and *XBee-API* [30] libraries provide basic routines to let the Arduino board and the Raspberry device, respectively, interact with XBee radios.



(a)



(b)

Figure 8: Software modules implemented in the M2M Gateway (a) and in the Arduino board (b).

6 Performance assessment

In this Section we report some experimental results obtained over the deployed test-bed to assess the performance of the proposed *lNDN* modules and the resulting integration with the ETSI M2M system.

Table 2: M2M devices: memory footprint

Library	ROM	RAM
lNDN-arduino	12.096 KB	795 bytes
Arduino-μIPv6Stack	34.972 KB	3.690 KB

Table 3: M2M devices: processing footprint

Library	Action type	Time[μsec]
lNDN-arduino	name matching success	0.358
	name matching failure	0.266

Since a major contribution of our proposal is the support of NDN group-based communications in the M2M Area Network, the experiments also compare the data collection performance when using the legacy NDN one Interest-one-Data exchange and the multi-source data retrieval based on the designed name conventions.

For the sake of completeness, an IP-compliant solution has been also assessed as benchmarking. To this purpose, the Arduino boards were equipped with an IPv6 stack ported from the Contiki operating system, *Arduino-μIPv6Stack* [31]. Such implementation has been conceived with a reduced set of CoAP functionalities to keep an ultra small memory footprint, hence not natively deploying advanced features like group-based communications.

It is worth remarking that, according to [13], CoAP supports group-based communications. However, this requires additional signalling for the join/leave operation of each producer to the multicast group and for allocation of the group 's URI. Our NDN with wildcard Interest solution, instead, easily allows groups to be created without additional signalling, thanks to the expressiveness and granularity of the conceived naming scheme.

6.1 Implementation footprint

First, our analysis aims to measure the footprint of the software implemented in the extremely resource-constrained M2M devices and in the Gateway.

6.1.1 M2M devices analysis

Table 2 shows the memory, ROM and RAM, sizes of the *lNDN-arduino* binary and of the IP-based implementation.

Results show that the *lNDN-arduino* footprint is significantly lower compared to the *Arduino-μIPv6Stack* (more than halved, both in terms of ROM and RAM). This is mainly because of the huge IP-based stack load, i.e., CoAP/-UDP/6LoWPAN. Our proposal, instead, runs directly over the layer 2 (there is no need for the IP stack).

Such results, proving that the proposed approach is lightweight, confirm its viability and encourage us to go more deeply into its performance assessment.

Table 3 reports the processing time of common operations on the deployed *lNDN-arduino*, after the reception of the Interest: *(i) name matching success*, which includes the time needed to perform a successful name matching; *(ii) name matching failure*, which includes the time needed to perform the name check when no matching is found. It is worth noticing that both time values are almost negligible. In particular, in case of name matching failure, the device can soon go back to the sleep mode operation, so to save battery.

6.1.2 Gateway analysis

Results in Figure 9 report the CPU and RAM allocations at the Gateway for the deployed modules, i.e., GSCL, *lNDN-java*, XBee-API. For the sake of completeness, the footprint of the Java Virtual Machine (JVM), needed to run java software, is also reported.

It can be observed that most of the CPU consumption has to be ascribed to the XBee-API module, while the *lNDN-java* module incurs a negligible load, especially if compared to the heavier GSCL module.

Overall, the proposed solution can be easily deployed over the Raspberry device, although it is not a high-end node, which is expected to be dedicated to smart home monitoring tasks.

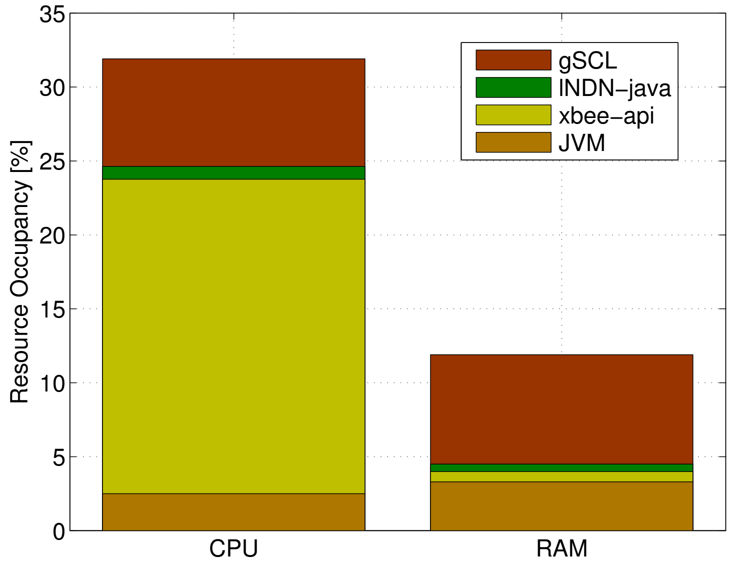


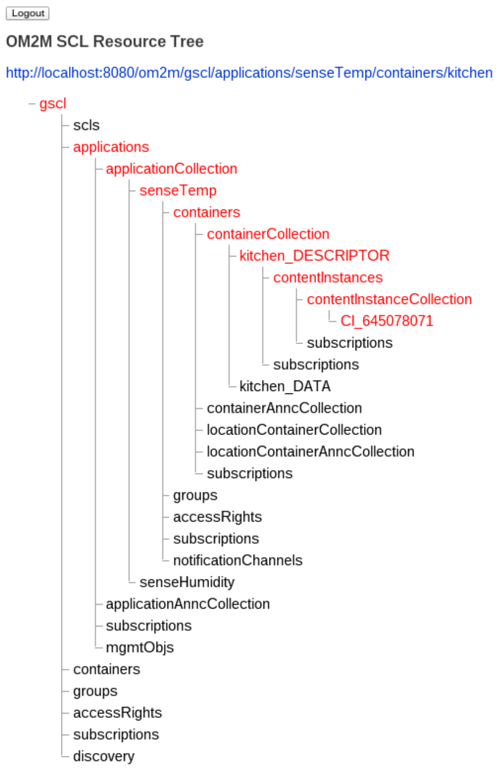
Figure 9: M2M Gateway: footprint of the implemented software modules.

6.2 Data delivery performance

To study the performance of data delivery in our integrated ETSI M2M-ICN system, we consider two different experiments:

- First, we focus on the data retrieval from the perspective of a remote application and show the information visualized via the OM2M web interface;
- Second, we deepen the analysis in the M2M Area Network by considering group-based communications and evaluate two performance metrics: the data collection time and the energy consumption at the M2M devices.

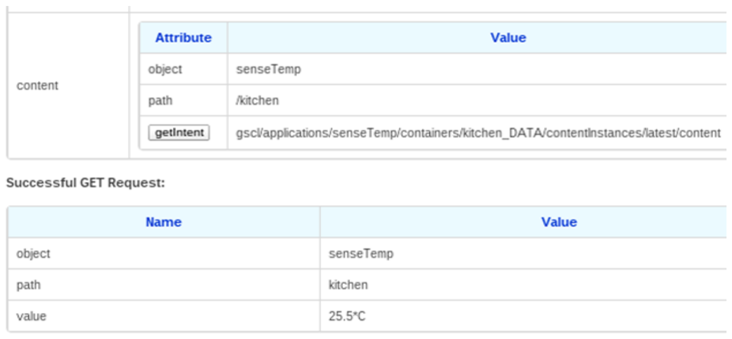
In our scenario, the GSCL running on the home Gateway registers with the NSCL and makes accessible to the NAs a set of sensing applications able to report environmental parameters. Specifically, the house consists of *five* rooms (i.e., kitchen, bedroom1, bedroom2, bathroom, living), physically coincident with our laboratory facilities, and each one holds an M2M device provided with a temperature and a humidity sensor. Therefore, each M2M device hosts the so-called *senseTemp* and *senseHumidity* applications, following the DA namespace conventions described in Section 4.2. The resources exposed by the GSCL are identified by URIs, in the ETSI M2M format, and represented in a hierarchical tree.



```
Logout
OM2M SCL Resource Tree
http://localhost:8080/om2m/gscl/applications/senseTemp/containers/kitchen

- gscl
    scls
    applications
        applicationCollection
            senseTemp
                containers
                    containerCollection
                        kitchen_DESCRIPTOR
                            contentInstances
                                contentInstanceCollection
                                    CI_645078071
                                subscriptions
                            subscriptions
                        kitchen_DATA
                    containerAnncCollection
                    locationContainerCollection
                    locationContainerAnncCollection
                    subscriptions
                groups
                accessRights
                subscriptions
                notificationChannels
            senseHumidity
        applicationAnncCollection
        subscriptions
        mgmtObjs
    containers
    groups
    accessRights
    subscriptions
    discovery
```

Figure 10: GSCL resource tree visualized with the OM2M web interface.



content	Attribute	Value
	object	senseTemp
	path	/kitchen
	getIntent	gscl/applications/senseTemp/containers/kitchen_DATA/contentInstances/latest/content

Successful GET Request:

Name	Value
object	senseTemp
path	kitchen
value	25.5°C

Figure 11: The requested temperature value (25.5°C) is visualized via the OM2M web interface.

After registering to the NSCL, an NA discovers the services hosted at the GSCL.

In the first experiment, we assume that the GSCL exposes only the sDAs (senseTemp, senseHumidity) in the kitchen, as shown in Figure 10. The NA sends a *Retrieve* message to obtain the temperature value, which is visualized as a response data in the OM2M web interface, as shown in Figure 11.

The processing time required by the GIP to translate the *Retrieve* message into an NDN Interest (and vice-versa) is negligible compared to the retrieval delay, thanks to the similarities between the hierarchical names. The measured retrieval delay consists of two main factors: *(i)* the round-trip-time between the Gateway and the Desktop-PC hosting the NA (about $100ms$) and *(ii)* the round-trip-time between the Gateway and the M2M device (about $140ms$).

In the second experiment, centred around the multi-source data retrieval, we refer to the *groups* resource in the GSCL hierarchical tree. The NA sends a *Retrieve* message to obtain *all* the available temperature data from the house. When the collection is completed, the Gateway sends back the retrieved content. Again, from the user high-level perspective, the time to process the request and build the ETSI M2M Response packet is almost negligible compared to the retrieval delay. We compare the communication in the M2M Area Network in the following NDN cases:

- The *legacy NDN Interest/Data* exchange, where an Interest packet retrieves a single Data packet; this implies that Interests must be sent by the Gateway in a round-robin scheme *for each named resource*, individually.

- The proposed *multi-source* retrieval, which retrieves all the temperature data with a single Interest, thanks to the proposed name conventions.

Results are also reported for the IP-based protocol suite (COAP/UDP/6LoWPAN), with the *Arduino-µIPv6Stack* installed on M2M devices. Being our study fo-

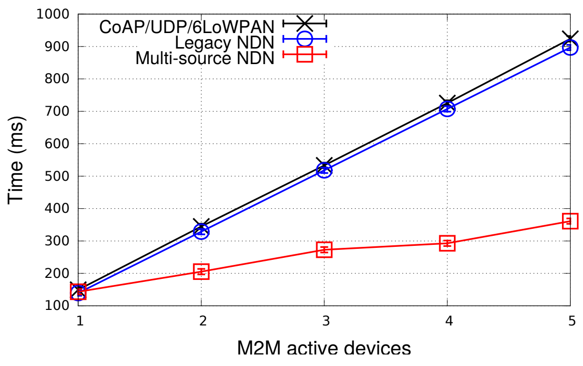


Figure 12: Average data collection time when varying the number of active M2M devices.

cused on assessing the preliminary viability and deployability of the proposed M2M solution, security features are not implemented in neither case.

We vary the number of M2M devices (i.e., the temperature sensors acting as producers) available in the test-bed, from 1 to 5, and repeat the experiment to get values averaged over 50 experimental runs (95% confidence intervals are reported in the plots).

Figure 12 reports the mean collection time required for the Gateway to obtain the Data replies from all the M2M devices, starting from the request transmission(s). Such a time includes *(i)* the transmission delay of both Data request (i.e., the Interest in NDN) and response (i.e., the Data in NDN) packets over the XBee radio interface, *(ii)* the communication delay between the Raspberry/Arduino devices and the Xbee shield[3], and *(iii)* additional processing times at the Arduino board.

First, we can observe that there are no remarkable differences between the legacy NDN and the COAP/UDP/6LoWPAN solutions. In fact, similarly to the legacy NDN, the devices implementing *Arduino-μIPv6Stack* are sequentially polled by the Gateway through request/response. In both cases, the request/response packets carry the same URI, i.e., the name that identifies the requested data. Moreover, since the 6LoWPAN header is compressed, COAP/UDP/6LoWPAN packets are only a few bytes larger than NDN packets. Therefore, the mildly better performance of legacy NDN w.r.t. the IP-based solution can be ascribed to the slightly smaller packet transmission and processing times and, as previously discussed, to the lower implementation footprint compared to the *Arduino-μIPv6Stack*.

In presence of a single source, the performance of legacy and *multi-source* NDN coincides with the CoAP-based solution. Overall, when varying the number of producers, the delay increases and more heavily when each single source needs to be addressed with an Interest packet. Specifically, with five producers,

[3]The communication occurs through standard Arduino serial commands, with a maximum baud rate of 115200 bps.

the collection time values get significantly lower for the proposed multi-source retrieval compared to the legacy NDN retrieval. The available temperature data, in fact, are collected in around $300ms$ with multi-source retrieval, while the legacy NDN takes $900ms$. This is due to the fact that the legacy NDN requires an individual polling of the sources, and a single Interest-Data exchange takes about $140ms$. Processing operations at the Arduino are indeed slow, due to the intrinsic hardware constraints of the device. Vice versa, the multi-source Interest allows the Arduino devices to process *in parallel* the single request, and to build in parallel the correspondent answers. Data packets are then sent with a short deferring time just to avoid the collision.

In addition to a reduction in the data collection time, the multi-source retrieval further benefits from a reduced network traffic and load on the constrained devices. The devices, in fact, will receive and process only one request from the Gateway, instead of being forced to receive multiple (useless) Interests, which will be discarded since they target a different temperature resource.

To capture the beneficial effect of this reduced traffic and processing load, we compute the energy consumption at the Arduino devices. To enable measurements, the Arduino is powered with a stabilized power supplier, configured to provide a 9 V output voltage (the same as the nominal value of the Arduino board). A 5 1/2 digit multimeter is also inserted, as illustrated in Figure 13, in order to measure the current passing through the board. The multimeter is connected via an USB interface to a laptop, where a Labview application collects measurements of current.

Figure 14 shows the average energy consumption for a target Arduino device during the collection process.

Since the energy consumption is directly related to the time spent in transmitting, processing and receiving packets, the performance in terms of energy consumption follows the trend of the data collection time. The legacy NDN solution slightly outperforms the IP-based implementation, while, in presence of multiple devices, the multi-source retrieval is more energy efficient than the legacy NDN.

The difference between legacy and multi-source NDN is again mainly due to the fact that the legacy NDN generates an Interest for any resource that must be retrieved. Therefore, each Arduino is involved in the reception and processing of multiple Interests, with consequent energy waste. With multi-source NDN, instead, a multi-source Interest is sent only once, thus reducing the load (and energy consumption) on the constrained devices.

7 Conclusion

In this paper, we presented a communication framework that leverages the ICN paradigm for local M2M communications and enables global access to the resources through the ETSI M2M architecture.

ICN and, in particular, the NDN instantiation, exhibits unique features that well suit the requirements of M2M communications (e.g., information-centric

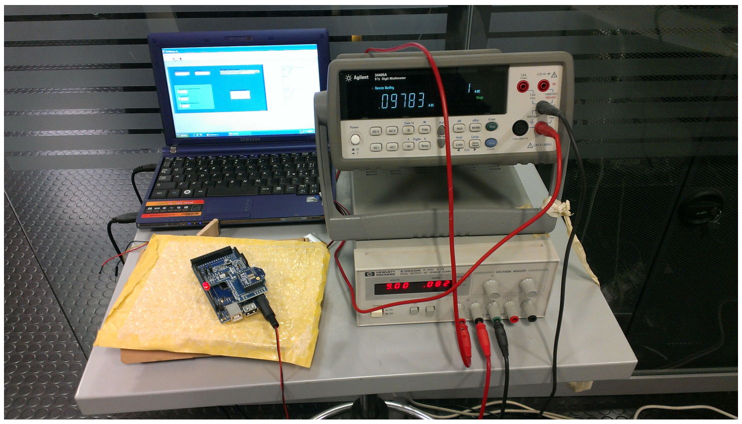


Figure 13: Setup for our power measurements: the laptop on the left hosts the application for evaluating performance of the M2M device.

applications, group-based communications). Moreover, the hierarchical naming scheme is quite similar to the one devised within the ETSI M2M reference architecture, hence facilitating the interoperability with remote entities. We made the best of such factors and our work capitalized on the design of an ICN-based framework to support lightweight interactions in an M2M Area Network with resource-constrained devices.

We have assessed the viability of the proposed framework through the deployment of a test-bed mimicking a home network with off-the-shelf low-cost devices equipped with an IEEE 802.15.4 interface and embedding environmental sensors.

The OM2M open source platform has been used to implement the ETSI M2M system. It has been integrated with the ICN-capable Home Area Network through an interworking proxy capability deployed at the home Gateway.

Achieved experimental results confirmed that the proposed solution provides a good trade-off in terms of simplicity of implementation and performance. Moreover, the proposed solution for group-based communications outperforms the legacy NDN one Interest-one Data exchange and the IP-based implementation, and it is flexible enough to group M2M devices according to their physical location or the sensed parameter, thanks to an expressive naming scheme.

References

References

[1] J. Kim, J. Lee, J. Kim, J. Yun, M2M Service Platforms: Survey, Issues, and Enabling Technologies, Communications Surveys & Tutorials, IEEE 16 (1) (2014) 61–76.

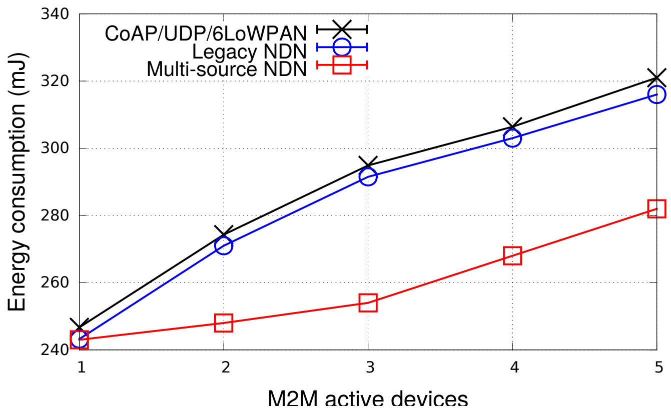


Figure 14: Average energy consumption per received request per device when varying the number of active M2M devices.

[2] C. Antón-Haro, T. Lestable, Y. Lin, N. Nikaein, T. Watteyne, J. Alonso-Zarate, Machine-to-Machine: an emerging communication paradigm, Transactions on Emerging Telecommunications Technologies 24 (4) (2013) 353–354.

[3] ETSI TS 102 690 v2.1.1: Machine-to-Machine Communications (M2M); Functional Architecture (2013).

[4] J. Swetina, G. Lu, P. Jacobs, F. Ennesser, J. Song, Toward a Standardized Common M2M Service Layer Platform: Introduction to oneM2M, Wireless Communications, IEEE 21 (3) (2014) 20–26.

[5] M. B. Alaya, Y. Banouar, T. Monteil, C. Chassot, K. Drira, OM2M: extensible ETSI-compliant M2M service platform with self-configuration capability, Procedia Computer Science 32 (2014) 1079–1086.

[6] E. Borgia, The Internet of Things vision: Key features, applications and open issues, Computer Communications 54 (2014) 1–31.

[7] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, B. Ohlman, A Survey of Information-Centric Networking, Communications Magazine, IEEE 50 (7) (2012) 26–36.

[8] M. Amadeo, C. Campolo, A. Molinaro, G. Ruggeri, Content-centric wireless networking: A survey, Computer Networks 72 (2014) 1–13.

[9] K. Katsaros, W. Chai, N. Wang, G. Pavlou, H. Bontius, M. Paolone, Information-centric networking for machine-to-machine data delivery: a case study in smart grid applications, Network, IEEE 28 (3) (2014) 58–64.

[10] W. Shang, Q. Ding, A. Marianantoni, J. Burke, L. Zhang, Securing Building Management Systems using Named Data Networking, Network, IEEE 28 (3) (2014) 50–56.

[11] J. Zhang, Q. Li, E. M. Schooler, iHEMS: an information-centric approach to secure home energy management, in: IEEE SmartGridComm, 2012.

[12] ETSI TS 102 921 v1.1.1: Machine-to-Machine Communications (M2M); mIa, dIa and mId Interfaces (2012).

[13] Z. Shelby, et al., Constrained Application Protocol (CoAP), draft, RFC 7252, The Internet Engineering Task Force–IETF.

[14] C. Bormann, M. Ersue, A. Keranen, Terminology for Constrained-Node Networks, Internet Engineering Task Force (IETF), RFC 7228.

[15] E. Baccelli, C. Mehlis, O. Hahm, T. C. Schmidt, M. Wählisch, Information Centric Networking in the IoT: experiments with NDN in the wild, in: Proceedings of the 1st international conference on Information-Centric Networking, (ACM ICN), 2014.

[16] M. Amadeo, C. Campolo, A. Iera, A. Molinaro, Named Data Networking for IoT: an Architectural Perspective, in: IEEE EuCNC, 2014.

[17] M. Amadeo, C. Campolo, A. Molinaro, Internet of Things via Named Data Networking: The support of push traffic, in: Network of the Future (NOF), 2014 International Conference and Workshop on the, IEEE, 2014, pp. 1–5.

[18] R. Ravindran, T. Biswas, X. Zhang, A. Chakraborti, G. Wang, Information-centric Networking based homenet, in: IFIP/IEEE ManFI Workshop, 2013.

[19] W. Shang, Q. Ding, A. Marianantoni, J. Burke, L. Zhang, Securing Instrumented Environments over Content-Centric Networking: the Case of Lighting Control and NDN, in: IEEE Infocom NOMEN Workshop, 2013.

[20] L. A. Grieco, M. Ben Alaya, T. Monteil, K. Drira, Architecting Information Centric ETSI-M2M Systems, in: Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on, IEEE, 2014, pp. 211–214.

[21] L. Zhang, et al., Named Data Networking (NDN) Project, Tech. Rep. NDN-0001, PARC (October 2010).

[22] M. Meisel, V. Pappas, L. Zhang, Ad hoc networking via named data, in: Proceedings of the fifth ACM international workshop on Mobility in the evolving internet architecture, 2010, pp. 3–8.

[23] M. Amadeo, C. Campolo, A. Molinaro, Multi-source data retrieval in IoT via Named Data Networking, in: Proceedings of the 1st international conference on Information-Centric Networking, (ACM ICN), 2014.

[24] Raspberry Pi, [on-line] `http://www.raspberrypi.org/`.

[25] XBee tecnology, [on-line] `http://www.digi.com/xbee/`.

[26] IEEE standard for local and metropolitan area networks–part 15.4: Low-rate wireless personal area networks (LR-WPANs) amendment 1: Mac sub-layer, IEEE Std 802.15.4e-2012 (Amendment to IEEE Std 802.15.4-2011) (2012) 1–225.

[27] Arduino, [on-line] `http://www.arduino.cc/`.

[28] J. P. Meijers, et al., A Two-Tier Content-Centric Architecture for Wireless Sensor Networks, in: IEEE ICNP, 2013.

[29] Xbee-arduino, [on-line] `https://code.google.com/p/xbee-arduino/`.

[30] Xbee-api, [on-line] `https://code.google.com/p/xbee-api/`.

[31] Arduino-ipv6stack, [on-line] `https://github.com/telecombretagne/Arduino-IPv6Stack`.