An efficient resource naming for enabling constrained devices in SmartM2M architecture

Elvis Vogli, Mahdi Ben Alaya, Thierry Monteil, Luigi Alfredo Grieco, Khalil Drira

An efficient resource naming for enabling constrained devices in SmartM2M architecture

Elvis Vogli[1, 2], Mahdi Ben Alaya[2, 3], Thierry Monteil[2], Luigi Alfredo Grieco[1] and Khalil Drira[2]

[1] DEI, Politecnico di Bari,via Orabona 4, 70125, Bari, Italy ,
[2]CNRS, LAAS,7 avenue du Colonel Roche,F-31400 Toulouse, France,
[3]Univ. deToulouse,INSA,LAAS,F-31400 Toulouse,France,
Email: {elvis.vogli, alfredo.grieco}@poliba.it; {evogli, ben.alaya, monteil, khalil}@laas.fr

***Abstract*—The Internet of Things (IoT) paradigm envisions an interconnected world where things can communicate with each other, transmit state information and execute smart tasks. The number of devices connected in Internet has been growing exponentially in the past years and the Machine-to-Machine plays a major role in enabling IoT paradigm while it is expected that in the near future billion of devices will be connected. Till now many vertical solutions in the M2M domain exist. To bridge this gap , the European Telecommunications Standards Institute (ETSI) has defined SmartM2M a horizontal service layer that separates the communication part from the application domain and guaranteeing interoperability among different technologies. Each device independently from the technology should register and post data to the service layer, which makes them available to applications in a seamless way. However the integration with very constrained devices like a 6LoWPAN based network is not straight forward. In this paper at first it is analyzed the problem of the hierarchical Uniform Resource Identifier (URI) in the SmartM2M standard specifications. Then a new non hierarchical resource structure is presented which enables the use of very short overall URI for constrained devices.**

***Keywords*—6LoWPAN, CoAP, ETSI, IoT, M2M, SmartM2M, REST**

I. Introduction

The Internet of Things (IoT) represents, in the context of networking and services, one of the most relevant innovations of the third millennium [1], [2]. The vision of this new paradigm is a world-wide network with a tremendous amount of heterogeneous interconnected objects where all the types of real-world physical elements (sensors, actuators, personal electronic devices, or home appliances) are able to autonomously interact with each other. The IoT defines in other words a cyber-physical system that enables many advanced applications, either in domestic (i.e, domotics, assisted living, e-health etc.) or in industry domain (i.e, automation, logistics, manufacturing, intelligent transportation etc.). At the same time, the IoT entails an unprecedented gamut of new technical challenges, only partially addressed so far [3], [4].

In this context, low power and short range wireless communication technologies play a key role since they can enable the creation of spontaneous, pervasive, and energy efficient communication networks of smart nodes. Such technologies are commonly adopted in Wireless Sensor Networks (WSNs) and Machine-to-Machine (M2M) systems [5]. The standard with the longest-standing impact is IEEE 802.15.4 [6], which defines a low-power Physical (PHY), and a Medium Access Control (MAC) layer, which has been the foundation of ZigBee [7]. This standard has been extended by its new IEEE 802.15.4e amendment [8]. Among other features, it introduces the Time Slotted Channel Hopping (TSCH) mode to increase the reliability and the energy efficiency of short range wireless communications in noisy environments [9]. Moreover various Internet Engineering Task Force (IETF) Working Groups (WGs) facilitate the integration of low-power wireless networks into the Internet and define a complete IoT stack for WSN [10]. The IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) WG [11] defined an adaptation layer that compresses the IPv6 header to make it suitable for constrained networks. The IPv6 over Networks of Resource-constrained Nodes (6lo) WG focuses on the work that facilitates IPv6 connectivity over constrained node networks. The Routing Over Low power and Lossy networks (ROLL) WG has defined the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) a routing protocol for WSN[12]. Finally the Constrained RESTful Environments (CORE) WG has defined The Constrained Application Protocol (CoAP) a lightweight protocol which enables Representational State Transfer (REST) architecture in constraint devices.

Many other component-level standards exist, addressing different radio interfaces or different networking choices. Each choice is optimized for a particular application scenario therefore many vertical M2M solutions have been designed independently. This inevitably hinders a large-scale M2M deployment. Therefore the largest part of the challenge facing the M2M actors is to transform vertical silos into a set of easily developable and incrementally deployable applications. This in term has brought to the deployment of horizontal platforms. What is meant by "horizontal" is a coherent framework valid across a large variety of business domains, networks, and devices. This horizontal middleware layer makes the data independent from the network access, which in term facilitates the interoperability among different network technologies.

To this end, the European Telecommunications Standards Institute (ETSI) has been working to define a set of specifications which provide a RESTful architecture able to standardize the way heterogeneous devices can offer services and access to them seamlessly [13]. Such platform facilitates the deployment of vertical applications and boosts innovation across industries for an effective interoperability. Therefore a device which uses REST application protocol like HyperText Transfer Protocol

(HTTP) or CoAP should not have problems interacting with the ETSI M2M platform. This however is not straight forward when it comes to constrained devices like the 6LoWPAN sensor/actuator devices.

In this paper we analyze the application registration phase of 6LoWPAN constrained devices in an ETSI SmartM2M platform and the impact of the URI is highlighted. Considering that hierarchical resource names produce long URIs, an efficient and non hierarchical resource structure was proposed and evaluated. In particular, it is demonstrated that the URIs in the proposed solution are shorter with fixed length and the number of messages required in the initialization phase is smaller, making this solution better suited for constrained devices.

The rest of the paper is organized as follows. In Sec. II,at first an overview on ETSI SmartM2M architecture and the resource registration phase is given, than it is described the the protocol stack of the constrained devices. In Sec. III it is described a non hierarchical schema for M2M resources to reduce the overall length of URIs. Sec. IV are evaluated the proposed solutions in comparison with the standardized SmartM2M approach. Finally, Sec. V gives the conclusions and future works

II. An Overview on ETSI M2M standard

According to the Global Standards Collaboration Machine-to-Machine Task Force, more than 140 organizations around the world are involved in M2M standardization. A considerable effort is done by ETSI to decrease M2M market fragmentation by defining a horizontal service platform for M2M interoperability. The proposed solution provides a RESTful Service Capability Layer (SCL) [14] accessible via open interfaces to enable developing services and applications independently of the underlying network. A SCL interfaces the network access from the application domain. Each device has to register the resources in the SmartM2M standard platform which in turn can be accessed from the applications from SmartM2M platform in a seamless way.

Figure 1 describes the SmartM2M functional architecture. An SCL can be deployed on an M2M Network (NSCL), a Gateway (GSCL), or a Device (DSCL). It provides several service capabilities to enable machine registration, synchronous and asynchronous communication, resource discovery, access rights management, group broadcast, etc.

An M2M Device runs applications using the SCL. It can connect directly to the Network Domain via the Access network and may provide service to other devices connected to it that are hidden from the Network Domain. It can also be connected to the Network Domain via a Gateway through a Local Area Network. A Gateway also runs M2M Applications using the SCL, and can act as a proxy between local devices and the network domain.

The access network allows M2M devices and gateways to communicate with the Core Network. The SCL provides functions that can be shared by different Applications. Network Management Functions enables to manage the Access and Core Networks, such as Provisioning, Supervision, and Fault Management. M2M Management Functions consist of all the functions required to manage the SCL in the Network Domain.

Three reference points [15] based on open APIs are specified: mIa, dIa, and mId. The mIa reference point allows a Network Application (NA) to access the NSCL. The dIa allows a Device or Gateway Application (D/GA) to access the D/GSCL. The mId reference point allows a D/GSCL to access the NSCL. These interfaces are defined in a generic way to support a wide range of network technologies and protocols to enhance interoperability.

SmartM2M adopts a RESTful architecture style. Each SCL contains a standardized resource tree where the information is stored. A resource is uniquely addressable via a URI, and has a representation that can be transferred and manipulated with verbs (e.g. retrieve, update, delete, and execute).

An SCL resources tree supports different kind of resources. The “sclBase” resource describes the hosting SCL, and is the root for all other resources within the hosting SCL. The “scl” resource stores information related to distant SCLs, residing on other machines, after successful mutual authentication. The “application” resource stores information about the application after a successful registration on the hosting SCL. The “container” resource acts as a mediator for data buffering to enable data exchange between applications and SCLs. The “contentInstance” resource represents a data instance in the container. The “accessRight” resource manages permissions and permissions holders to limit and protect the access to the resource tree structure. The “group” resource enhances resources tree operations by adding the grouping feature. The “registration” resource allows subscribers to receive asynchronous notification when an event happens such as the reception of new sensor event or the creation, update, or delete of a resource. The “announced” resource contains a partial representation of a resource in a remote SCL to simplify discovery request on distributed SCLs. The “discovery” resource acts as a search engine for resources. The collection resource groups common resources together. A partial view of the resource tree is shown in Figure 2.

In this work the OM2M project[16], which provides a complete open source implementation of the SmartM2M standard including both the server and the gateway part will be used to validate the proposed approaches. The main concern of this contribution is focused in connecting constrained devices (i.e., 6LoWPAN based wireless sensor network) to the Gateway over the dIa interface.

A. Device Application Gateway SCL registration

Each Device Application (DA) should register to its Gateway SCL. The registration phase can be divided in three parts. In the first part, an “Application” resource is created. In the second part it is created a description “container” where are stored descriptive information related to the application. Finally in the last part there is created a data “container” where are stored the application data (i.e., sensor or actuator data).

In Table I there are described the messages that an example application called *TEMPERATURE* will send to the G/NSCL as defined in ETSI SmartM2M standard. After each POST message a new resource in the resource tree will] be created. In particular there are created the highlighted resources shown in Fig. 2. The first message sent will create *TEMPERATURE* application resource

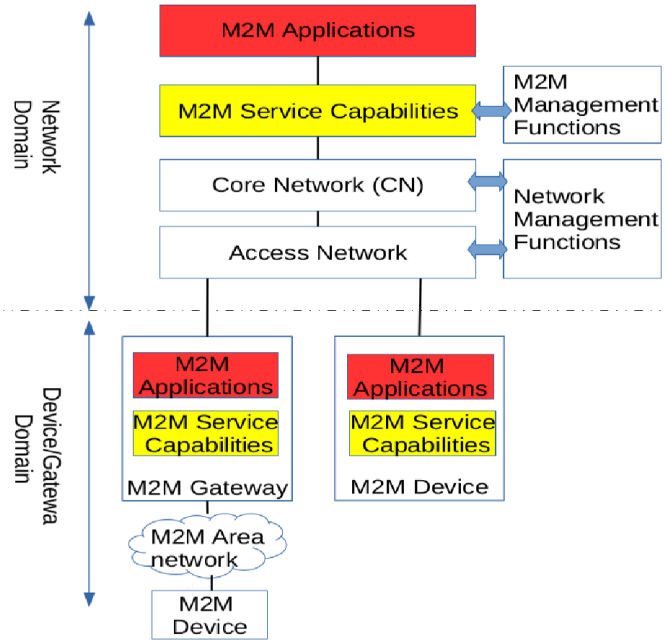


Fig. 1: SmartM2M high level architecture.

accessed through the */gscl/applications* URI. The second message will create a *DESCRIPTOR* container with URI */gscl/applications/TEMPERATURE/containers* which will contain a content instance created from the third message. The third message will provide descriptions and semantics related to the created application accessed through the URI */gscl/applications/TEMPERATURE/containers/DESCRIPTOR-/contentInstances*. The fourth message will create a *DATA* container resource with URI */gscl/applications /TEMPERATURE/containers/DATA* which will hold content instances created from the fifth message. And finally, the fifth will send to the SCL the sensor data which are stored in */gscl/applications/TEMPERATURE/containers/DATA/content-Instances*. The last message will be sent from the DA each time it needs to communicate a new sensor value.

B. Overview on the WSN stack

The protocol stack depicted in Figure 3(a) represents one of the most promising stacks for IoT. Above the efficient 802.15.4(e) MAC the Internet connectivity is obtained thanks to 6LoWPAN which enables IPv6 addressing in constrained devices. In addition the CoAP application protocol provides a RESTful protocol for constrained devices. Despite the fact that this stack is designed to be implemented in M2M scenarios their integration is not a simple task. A major constraint is imposed from the packet length where the MAC layer limits the maximum frame length to 127 bytes.

If we consider the headers of each of the protocols in the depicted stack, in the application layer will barely be available 50 Bytes. Neither one of the messages depicted in Tab. I) can be transmitted in a single frame, therefore fragmentation mechanisms should be used. The use of fragmentation in the link layer or at the IP layer to enable the transport of larger representations is possible, but the fragmentation and

TABLE I: POST messages requested for the registration of a Device application in the GSCL. Example in the OM2M implementation.

Msg.	Content	Size
Msg. Nr. 1 - Create an application Resource (TEMPERATURE)		
POST	/gscl/applications	18
Payload	<om2m:application xmlns:om2m="http://uri.etsi.org/m2m" appId="TEMPERATURE"> <om2m:aPoCPaths> <om2m:aPoCPath> <om2m:path> coap://[IPv6_Prefix::IPv6_Suffix]:5683 </om2m:path> </om2m:aPoCPath> </om2m:aPoCPaths> </om2m:application>	234
Msg. Nr. 2 - Create an DESCRIPTOR container		
POST	/gscl/applications/TEMPERATURE/containers	41
Payload	<om2m:container xmlns:om2m="http://uri.etsi.org/m2m" om2m:id="DESCRIPTOR"/>	75
Msg. Nr. 3 - Create a content instance in the DESCRIPTOR container		
POST	/gscl/applications/TEMPERATURE/containers/DESCRIPTOR /contentInstances	69
Payload	<obj> <str name="type" val="Temperature_Sensor"/> <str name="location" val="Home"/> <op name="getValue" href="gscl/applications /TEMPERATURE/containers/DATA/contentInstances /latest/content" is="retrieve"/> <op name="getDirectValue" href="gscl/applications /TEMPERATURE/sl" is="retrieve"/> </obj>	260
Msg. Nr. 4 - Create an DATA container		
POST	/gscl/applications/TEMPERATURE/containers	41
Payload	<om2m:container xmlns:om2m="http://uri.etsi.org/m2m" om2m:id="DATA"/>	70
Msg. Nr. 5 - Create a content instance in the DATA container		
POST	/gscl/applications/TEMPERATURE/containers/DATA /contentInstances	63
Payload	<str name="data" val="XYZ"/>	28

successively the reassembly is a burden on the lower layers of resource constrained devices and can be more efficiently managed in the CoAP layer.

In CoAP there are defined Block options which provide a way to transfer large representations in block-wise fashion. Using these options, larger resource representations can be fragmented and reassembled by CoAP independently of the lower layers as well as the above applications. In Figure 3(b) it is described the format of CoAP packets.

The header of fixed length of four bytes is followed from a token with size that can be up to 8 bytes. Than there are a set of several options. Finally there is the payload which should always be preceded from payload delimiter option (0xFF byte). Despite the existence of all the mechanisms necessary for the transfer of long representations the long URI length of the resources defined in the ETSI SmartM2M standard could become a relevant issue. The URI are sent as options in the CoAP header and should be present in each block. If it is very long there will not be possible to send the block in only

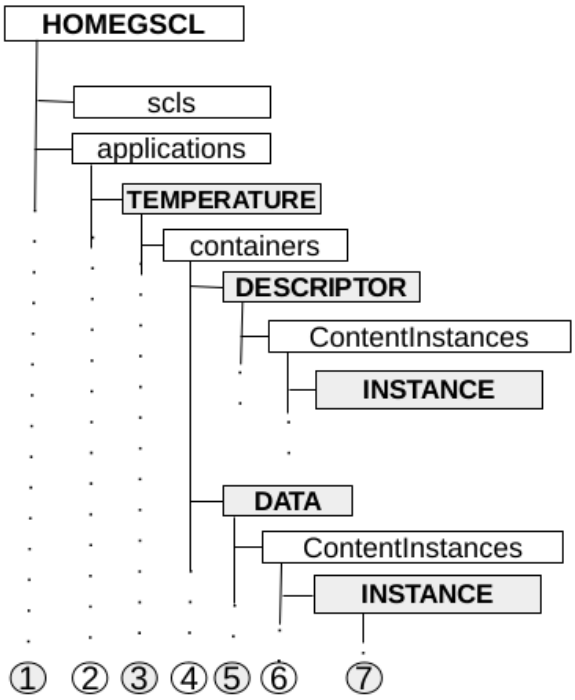
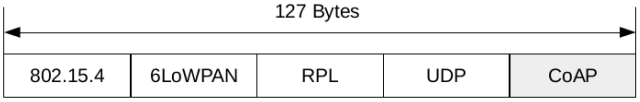


Fig. 2: ETSI URI resource tree structure (example).

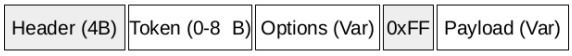
one message and the further datagram fragmentation will be needed.

In the case of the ETSI SmartM2M resource tree as it can be seen from the Figure 2 there are up to six different levels of the URI which can make the overall length particularly long. In some cases it not be possible to send in a single datagram CoAP blocks even if it is used the minimum block size (16 byte) due to the length of the URI.

This example clearly shows the necessity for optimal design of URIs in order to improve the efficiency in constrained devices.



(a) stack



(b) Coap Format

Fig. 3: Protocol stack and CoAP format.

III. Models

The SmartM2M standard has descriptive naming conventions for the resources such that the meaning of each resource can be clearly conveyed and understood. Some of these names are very long and in addition, the fact that the resource structure is hierarchical (i.e., tree like resource structure), as it can be verified from the Table I, can generate very long overall URIs. In the second version of the SmartM2M standard [17], in order to cope with constrained devices, there were introduced as an informative annex shorter naming conventions for resource and attributes. In particular resource names which are long only two byte and shorter attribute names are provided. The first one tend to minimize the URI length whether the second one have an impact on the payload data representation length. This short alias where defined appositely to cope with limitation imposed from constrained devices.

In Fig. 4 it is described the registration phase using the short resource names and structure provided from SmartM2M. The resource names provided from SmartM2M are respectively the resources in the level two, four and six in the resource tree of the Figure 2. Instead for the user defined resources have been considered three byte long names. For each request sent from the device application, the gateway will send back the resource created in the confirmation response. It can be clearly noticed that this approach provides shorter URIs compared to the standard descriptive names. However, despite the short length of the resources, the hierarchal resource structure sets a limit on the overall URI length. In the case there is retrieved a content instance which is in the seventh level, only for the path separator character (/) will be needed 7 bytes.

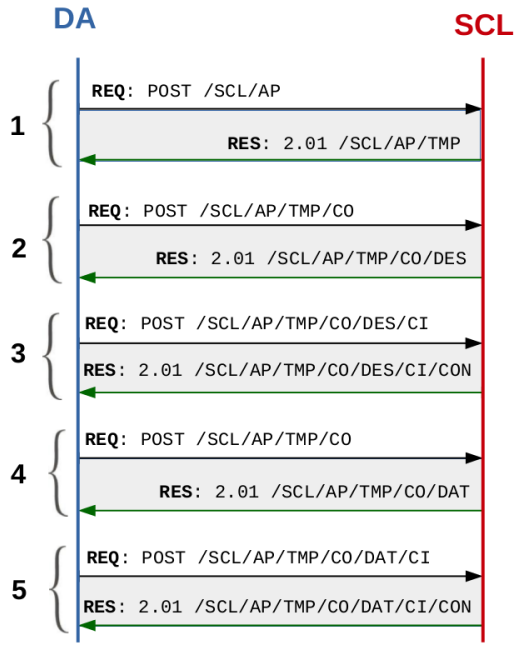


Fig. 4: Messages in the ETSI-M2M tree URI case.

A. Non Hierarchical URI model

Previously we have shown an example about SmartM2M with up to seven nested resources but in general this number can be bigger. To overcome the limitations imposed form the hierarchical resource structure defined in ETSI SmartM2M hereby we propose a flat non hierarchical resource structure.

We propose a new approach which limits the number of nested resources and therefore optimizes the overall URI length. In particular we distinguish two cases depending on the type of request message that will be used. In the first case, in the POST messages that will be used to create new resources the structure of the URI will be:

/<prefix>/<resource_id>/<suffix>

Each URI will have to three levels called “prefix”, “resource_id” and “suffix”. In order to create a new resource a POST request should be made to a URI having the aforementioned structure. The generic URI of the created resources will be:

/<prefix>/<resource_id>

The “prefix” of the resource created should be the “suffix” of the URI in the POST message used to create the resource. Moreover the resource URI will be the generic URI structure that will be used with the other REST methods (i.e., GET, PUT or DELETE).

In analogy to the SmartM2M example described previously in Fig. 5 there is shown the subscription phase of a device application to its gateway using the non hierarchical URI structure proposed. There are used two byte names for the “prefix“ and ”suffix“ (the same names defined in SmartM2M) and three byte names for created resources. In the first message a POST to /SB/SCL/AP will create the TMP application resource in /AP/TMP. In the response there will be included the created resource URI to inform the DA. In the second message it is created the descriptor container by a POST message sent on the ”container“ collection of the TMP application. The descriptor container resource will not be nested under the collection as in the SmartM2M case instead it will respect the generic URI defined. In this case the collection of the TMP application resource will become the prefix of the descriptor container resource. Therefore the descriptor resource path will be /CO/DES and it will have its own collections nested to it. Finally, in the third message it is created the description instance by POST message sent to the contentInstance collection with URI /CO/DES/CI. Again the new resource with containing the application description will be created in /CI/DES. In the fourth and fifth message there are created the data container (/CO/DAT) and the data content instance (/CI/dat).

In this schema, once it is decided the length of a resource, the overall URIs will always have the same length. As it can be seen the URIs used in POST messages have a length of ten bytes whether the URIs used for the other methods are seven byte. The length of the resource name should be long enough to be able to address all the resources in the gateway. In addition the prefix and the suffix represent collections. The number of this collections is really limited such that only one byte can be used to represent them all. Therefore the size can be furtherly reduced with two bytes becoming eight bytes for the POST messages and six bytes for the other methods.

IV. Evaluation

In the Tab. II and Tab. III there are compared the four different approaches which were described throughout the paper. The values shown in the table are calculated taking in consideration the following conditions.

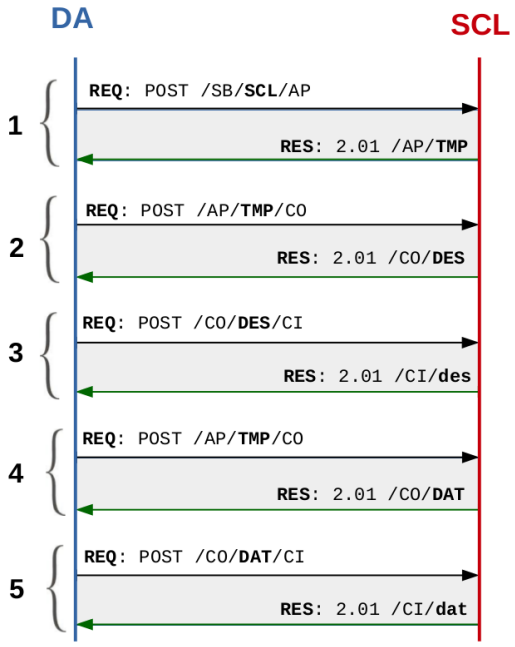


Fig. 5: Messages in the flat URI case.

- The user defined resources for the SmartM2M hierarchical resource scheme (i.e., the first, third, fifth and seventh level resources) and the “resource_id” in the proposed scheme are considered to have all the same length equal to three bytes.
- The number of messages and the number of blocks sent are calculated taking in consideration message representation described in Tab. I.
- It is considered a CoAP block-wise transmission with blocksize 16 bytes.
- It is considered that a CoAP packet bigger than 50 bytes will be fragmented into multiple datagrams.
- Only the messages required in the registration phase and the relative URIs are considered.

In the case of the SmartM2M with long resource names the overall number of block sent in registration phase will be doubled compared to the other cases. This happens because each CoAP fragment will split in two datagrams when the payload and CoAP header and options will exceed the length of the datagram payload.

In the Table II there are shown the length of the URIs used in the five POST messages sent during the registration phase and the length of the URI used in the GET message sent to retrieve the data of the resource created. The descriptive long names produce very long URI as it can be seen form the first raw of the Table II. In addition the URI lengths of the SmarM2M structured resources will be very long in the nested resources. The GET message will address resource which is in the seventh level therefore if we compare with the first POST it will be about three times bigger. This is happening only because of the hierarchical resource structure. The non

hierarchical resource structure proposed as it can be seen has a constant length among the POST messages and it has even a shorter length for the other methods.

In Table III there are evidenced the maximal URI sizes for the POST and the GET messages for the URI schema proposed and they are compared with both, the descriptive and the short named resource structure proposed defined in SmartM2M. In particular the non hierarchical names the corresponding URI is always 10 (8) bytes for POST (GET) messages whether the SmartM2M in the best case, where are used short names is 21 (25) bytes. In this situation the SmarM2M with short URIs requires the same number of messages as the proposed solutions but this scenario is not always possible. In particular if there are other options to be included in the CoAP header (i.e., like the authorization option in case there is enabled security) it is very easy that the messages will be fragmented even in the SmartM2M . In this situation it will be very easy that the CoAP length will become long enough to require the datagram fragmentation. In this context the non hierarchical URI proposed will be the choice where the spared bytes due to URI length reduction can be used to improve security or implement other functions without having to fragment the packets, or there can be used a CoAP block with a bigger size.

TABLE II: URI lengths in the POST messages used in subscription phase and the GET message to retrieve the data content instance created.

	POST Nr. 1	POST Nr. 2	POST Nr. 3	POST Nr. 4	POST Nr. 5	GET
SmartM2M Long names	18	41	69	41	63	71
SmartM2M Short names	7	14	21	14	21	25
Non-Hierarchical	10	10	10	10	10	7
Non-Hierarchical One Byte names	8	8	8	8	8	6

TABLE III: Maximal URI lengths and the number of frames required from each Device Application to register

	Maximal URI Length		
	POST	GET, PUT or DELETE	Nr. of blocks for registration
SmartM2M Long names	69	71	100
SmartM2M Short names	21	25	50
Non-Hierarchical	10	7	47
Non-Hierarchical One Byte names	8	6	47

V. Conclusion

Machine-to-Machine communication will boost up in the near future, but connecting constrained devices is not challenging. Beside the definition of lightweight protocols a very careful design of the resource names should be done to guarantee the correct use of the constrained devices. In this paper it was shown that the hierarchical resource names can produce very long URIs which can cause an increase of datagram fragmentation. For this reason a non hierarchical schema for resource naming was proposed and it was shown that the overall URIs will be short and suited for constrained devices. Despite the optimizations made in the URI length, there is still an elevated number of messages in the subscription phase. Considering the low data rate that characterizes constrained devices (like 6LoWPAN device), the overall time required for registration can be long. In addition to a careful URI design the message formats should be thoroughly optimized in order to decrease the overall number of messages.

References

[1] D. Miorandi, S. Sicari, F. D. Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497 – 1516, Feb. 2012.

[2] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787 – 2805, Oct. 2010.

[3] A. Whitmore, A. Agarwal, and L. D. Xu, "The internet of things—a survey of topics and trends," *Information Systems Frontiers*, pp. 1–14, Mar. 2014.

[4] S. Li, L. Xu, and S. Zhao, "The internet of things: a survey," *Information Systems Frontiers*, pp. 1–17, Jun. 2014.

[5] K. Chen and S. Lien, "Machine-to-machine communications: Technologies and challenges," *Ad Hoc Networks*, vol. 18, no. 0, pp. 3 – 23, Jun. 2014.

[6] IEEE std. 802.15.4, *Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)*, Standard for Information Technology Std., Jun. 2011.

[7] ZigBee Alliance. [Online]. Available: www.zigbee.org

[8] *802.15.4e-2012: IEEE Standard for Local and Metropolitan Area Networks – Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC Sublayer*, IEEE Std., Apr. 2012.

[9] L. Tang, K. Wang, Y. Huang, and F. Gu, "Channel characterization and link quality assessment of ieee 802.15.4-compliant radio for factory environments," *IEEE Transactions on Industrial Informatics*, vol. 3, no. 2, pp. 99–110, May 2007.

[10] M. Palattella, N. Accettura, X. Vilajosana, T. Watteyne, L. Grieco, G. Boggia, and M. Dohler, "Standardized Protocol Stack for the Internet of (Important) Things," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1389–1406, Mar. 2013.

[11] N. Kushalnagar, G. Montenegro, and C. Schumacher, *IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals*, RFC 4919, Internet Engineering Task Force RFC 4919, August 2007.

[12] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. P. Vasseur, and R. Alexander, *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*, RFC 6550, IETF RFC 6550, March 2012.

[13] O. Elloumi and C. Forlivesi, *M2M Communications: A Systems Approach*. John Wiley & Sons, Ltd, 2012.

[14] *ETSI TS 102.690 v1.1.1. Machine-to-Machine communications (M2M); Functional architecture*, Std., October 2011.

[15] *ETSI TS 102 921 v1.1.1. Machine-to-Machine communications (M2M); mIa, dIa and mId interfaces*, Std., February 2012.

[16] M. B. Alaya, Y. Banouar, T. Monteil, C. Chassot, and K. Drira, "Om2m: Extensible etsi-compliant {M2M} service platform with self-configuration capability," *Procedia Computer Science*, vol. 32, pp. 1079 – 1086, 2014.

[17] *ETSI TS 102 921 v2.1.1. Machine-to-Machine communications (M2M); mIa, dIa and mId interfaces*, Std., December 2013.