Service Discovery Protocols for Constrained Machine-to-Machine Communications

Berta Carballido Villaverde, Rodolfo De Paz Alberola, Antonio J. Jara, Szymon Fedor, Sajal K. Das and Dirk Pesch

***Abstract*— An emerging trend in many applications is to use resource-constrained wireless devices for machine-to-machine (M2M) communications. The observed proliferation of wireless embedded systems is expected to have a significant impact on future M2M applications if the services provided can be automatically discovered and accessed at runtime. In order to realize the decoupling of M2M applications and services, energy efficient service discovery mechanisms must be designed so as to minimize human intervention during configuration and management phases. However, many traditional service discovery protocols cannot be applied to wireless constrained devices because they introduce too much overhead, fail in a duty-cycled environment or require significant memory resources. To address this, either new protocols are being proposed or existing ones are adapted to meet the requirements of constrained networks. In this article, we provide a comprehensive overview of service discovery protocols that have been recently proposed for constrained M2M communications by the Internet Engineering Task Force (IETF). Advantages, disadvantages, performance and challenges of the existing solutions for different M2M scenarios are also analyzed.**

***Index Terms*—Service Discovery, Resource Discovery, M2M, Low Power Wireless Communications, Constrained Application Protocol (CoAP), Domain Name System (DNS).**

I. Introduction

Machine-to-machine (M2M) communications have been in existence for many years in the context of wired networks, for example Supervisory Control And Data Acquisition (SCADA) and Building Automation and Control Networks (BACnet). Due to the advent of new standards for low power wireless communications and the desire for mobile operators to find new sources of revenue, it is only recently that M2M wireless communications are gaining greater attention. Wireless low power devices are highly attractive in many scenarios due to the fact that they can be deployed in a wide range of applications and also easily retrofitted, thus significantly reducing installation costs. Moreover, many low power devices can work unattended for years, hence they are considered an excellent candidate for M2M applications such as building automation, where thousands of devices need to be deployed and maintained at a very low cost.

Up until recently low power networks architecture consists of a collection of low power devices reporting their gathered information to a sink which in turn disseminates the information to the outside world. This approach does not truly achieve direct M2M communications between any two peer devices. With the introduction of IP protocols into low power devices, the true potential of M2M communications can be realized in the sense that a unique device becomes discoverable and addressable by any other device remotely or locally [1]. Thus, low power or constrained M2M communications are considered as communications between two devices, where one of them is a constrained device, locally or remotely without the need for reporting to intermediary nodes. The advantage of this type of communications is that they enable applications where constrained sensors or actuators can report to servers (e.g. Smart Grid applications [2]) or be accessed by remote users (e.g. new generations of Home Automation applications [3][4]), which open up a complete new market based on globally accessible low power devices.

However, a significant challenge for low power M2M wireless communications to be a success in reality is to make these devices as autonomous as possible such that once deployed, they become "invisible". For this purpose, low power devices should require minimal human intervention at every stage of their operating life. This is important for large networks where configuration and maintenance of hundreds of devices often becomes a significant challenge, if not a burden, in particular when wireless devices join or leave the network as they move around the environment and/or their connectivity changes.

The traditional approach to reduce configuration and administration of network devices is with the help of service discovery mechanisms [1]. Specifically, discovery protocols allow devices and services to automatically become aware of the functionality and identity of other devices and services in the network without the need for human intervention. For example, a low power temperature sensing device may use a service discovery protocol to query other devices in the same

Manuscript received November 1, 2012.

B. Carballido Villaverde, R. De Paz Alberola, A. J. Jara and Szymon Fedor are with United Technologies Research Centre, Cork, Ireland (phone: +353214508440; e-mail: {carbalb, depazar, fedors}@utrc.utc.com, jara@ieee.org.

D. Pesch is with the Nimbus Centre for Embedded Systems Research, Cork Institute of Technology, Cork, Ireland (phone: +353214335171; e-mail: dirk.pesch}@cit.ie.

S. K. Das is with the Center for Research in Wireless Mobility and Networking, Department of Computer Science and Engineering, University Texas at Arlington, USA (phone: +18172727405; e-mail: das@cse.uta.edu).

area in order to determine which ones implement a heater switch service. This eliminates the need for an operator having to explicitly introduce this information to the sensing device. Several protocols such as Universal Plug and Play (UPnP) [6], Service Location Protocol (SLP) [7], JINI [8], or Salutation [9] have been proposed to enable service discovery between wireless/wired networked devices. However, since these protocols were not designed to be power efficient, they introduce significant overhead when adopted for constrained networks.

Here the term *constrained networks* refers to wireless networks composed of devices with limited power supply. This energy source limitation translates into the introduction of long inactivity periods (typically 99% of the time), reduced memory and processing capabilities (typically devices have 8 or 16 bit micro-controller with only ~10KB for data and ~100KB for program memory), and support for small frame sizes compared to the Internet infrastructure (typically 127 bytes as per IEEE 802.15.4 standard [Ref?] compared to 1280 bytes in IPv6).

In fact, in the context of constrained networks, the following additional design requirements have to be considered for service discovery protocols:

- Low overhead per control message and reduced number of message exchanges in order to reduce energy consumption and save communication bandwidth.
- Low memory and processing requirements so that the discovery mechanism can run even in highly resource-constrained nodes.
- Robust service provisioning to account for the high dynamics of the wireless communication channel and the unpredictable availability of battery powered devices[1].
- Interoperability with web applications and IP based back-end networks so that these low power devices do not operate in isolated “islands”.

Given the above design requirements, new mechanisms should be developed to enable service discovery in constrained networks. Along this line, recent advances in the standards development at the network and application layers, such as IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) [10] and Constrained Application Protocol (CoAP) [11] are making IP/Web enabled constrained networks a reality. CoAP, which is still work in progress of the Internet Engineering Task Force (IETF) Constrained RESTful Environments (CoRE) Working Group [12], is a RESTful[2] application layer protocol currently being designed to offer resource efficient, simple M2M communications to allow management and interaction among embedded devices. One of the requirements for CoAP design is defining how to use it to query or advertise a device’s description which may include name, list of its resources, and so on (REQ8 as per CoRE charter [13]). Thus, the IETF CoRE working group is currently developing several resource discovery mechanisms based on CoAP. In addition, other existing service discovery mechanisms such as Domain Name System Service Discovery (DNS-SD) [14] are also being considered by the IETF community for the same purpose. Besides the CoRE IETF efforts, non-IP based protocols such as Zigbee [15] and simplified versions of IP-based protocols such as SLP [7] have also been proposed for service discovery in constrained networks. As detailed later, there are still several challenges to overcome for service discovery within IP based low power networks in terms of overhead, reliability and scalability, among others.Surveys on service discovery protocols have been published in the literature [1], [16], [17], [18]. However, none of them focuses on constrained M2M communications. This motivates our work.

In this paper, we aim at providing a comprehensive review of service discovery mechanisms for constrained networks with a goal to provide a starting point of reference to gain further insights into these protocols. As a general introduction to the discovery concept, we first describe how service discovery protocols operate, detailing on different possible interactions common to all. Next we focus on those protocols that are currently being proposed by the IETF specifically for service discovery within constrained networks. Such protocols include CoAP resource discovery [19], CoAP Resource Directory (RD) [20] and DNS-SD, which can be based on multicast DNS (mDNS) [21], extended multicast DNS (xmDNS) [22] or unicast DNS. We then present a detailed tutorial on the operation of these protocols and underlying mechanisms, which are also evaluated in terms of overhead, discovery functionality, interoperability, scalability, reliability, robustness, required human interaction, and suitability for group communications. For the sake of completeness, a few additional service discovery protocols are also reviewed, with a focus on constrained networks. Finally, we describe some open source tools that may be used in testing the reviewed protocols.

The rest of this paper is structured as follows. Section II introduces the general concept of service discovery while Section III proposes taxonomy of state-of-the-art approaches for service discovery in constrained networks. Section IV details the operation of the protocols currently being proposed by the IETF for service discovery within constrained networks. Section V discusses advantages and disadvantages

[1] Note that this requirement conflicts with the low overhead requirement as changes in the network are likely to trigger more control message exchanges.

[2] REpresentational State Transfer (REST) is a software architecture style for distributed systems such as the World Wide Web. It has emerged as a predominant Web service design model.

of various approaches for different scenarios. Section VI presents open source tools that may be used to test service discovery protocols. It also discusses open issues and challenges. Finally, Section VII concludes the paper along with suggestions for future research directions.

II. Service Discovery Protocols: General Concepts

There are two broad classes of service discovery protocols: distributed and centralised. In a centralised approach, one or more service directories contain lists of services offered by the network devices which in turn use these directories as an intermediary to discover services. On the other hand, in a distributed approach, devices discover services by interacting with each other directly without any intermediary directory.

The main functionalities for a service discovery protocol during its lifecycle are publication, registration into the directory (if available), discovery (in terms of browsing), and the resolution. In addition, for those protocols based on a directory, several maintenance functionalities may be available to update, remove, and validate the entries. The following subsections describe the main functionalities of a service discovery protocol. Later these functionalities will be instantiated for IP-based service discovery protocols currently being considered by the IETF for low power (constrained) M2M communication networks.

A. *Publication*

Publication is commonly used by service discovery protocols that work in a distributed fashion, that is, whenever a directory is not available. Publication is a process whereby the devices announce/disseminate their own offered services. A device may only publish a subset of its offered services based on its requirements.

The *Publication* service commonly includes the following information illustrated in Figure 1.

- **Service type or class:** this is to allow filtering of services by search engines and clients depending on their interest. For example, *tempSensor* is a service type.
- **Service access:** this refers to mechanisms to access services, such as addresses and ports in IP networks (e.g., *[2001::11]:5683*), endpoints in ZigBee networks, or object IDs in BACnet networks.
- **Service name:** it differentiates a particular service from the allocated resource. For instance, if a particular device offers two similar services, say temperature sensors, each service should have a specific instance name to allow differentiation. Examples of service names are *temp001, temp002.*
- **Domain name:** the service domain and the device are included in case they are not exactly the same. An example domain for a building automation scenario may be *floor1.mybuilding.com.*
- **Service properties:** it provides a fine grained description of the type of service, the meta-data required by the search engine, or any other service related information given by the vendor. The service properties can be a list of key-value pairs or structured data such as XML. Examples of service properties may be *units=Celsius*.

B. *Registration*

Service descriptions can be stored by a network entity referred to as the directory. This directory keeps descriptions of services announced within its own local domain (or external domains for global directories). Registration is a process whereby the descriptions of services offered by network devices are stored (registered) into the directory to make these descriptions available for discovery within the respective domain.

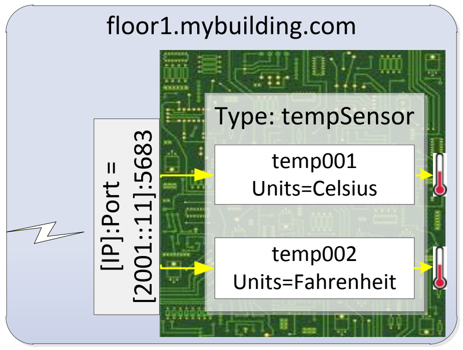

Figure 1: Example of Information Published by a Temperature Sensing Device

The establishment of the registration can be carried out through stateless or stateful mechanisms. In the stateless approach, the directory listens to service advertisements and populates its service database. The directory may also poll service descriptions directly from every device. Later on, the directory may answer to future queries on behalf of the original device (caching approach). On the other hand, in the stateful mechanism, an explicit registration is carried out directly from the device to the directory. In this case the constrained device must know the directory location or use a discovery mechanism such as a multicast request to find its location.

For the stateless solution, every received publication message can be utilized to collect service details, and accordingly update the registration entry. Updating the entry is required to keep the directory registrations coherent with the network status. Freshness can be managed through a lifetime value associated with every registered service such that outdated registrations can be removed from the directory.

For the stateful solution, since registrations are performed explicitly, additional protocol functionality needs to be available to maintain the directory entries. Such additional interfaces include:

- **Directory Discovery:** when the location of the directory is unknown, some discovery scheme must

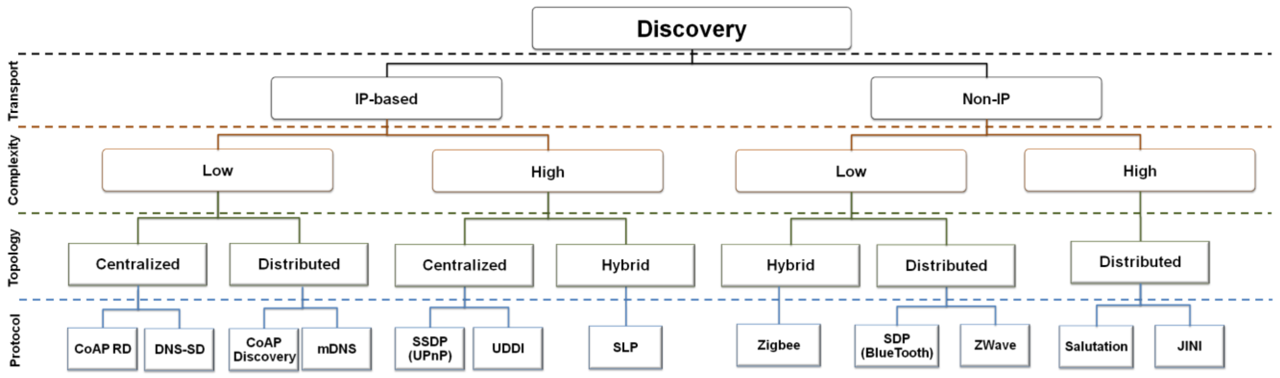

Figure 2: Taxonomy of Service Discovery Protocols

be available to locate it.

- **Registration Update:** a registration may be updated to reflect the latest service description offered by a service.
- **Registration Validation:** a directory may proactively check whether any stored registration is still valid by directly asking the server offering the service.
- **Registration Removal**: a device may unregister its service description from the directory when it is no longer available.

C. Discovery and Resolution

While the discovery refers to browsing a domain (neighborhood) to locate instances of some relevant service, the resolution relates to translating the discovered service instance to an accessible end-point address or host name. Note that even if some advertisement mechanism has been previously used, a network device may have to utilize the discovery when it has not overheard the required service advertisement. Browsing can be performed directly on the neighborhood, or on a centralized local directory, or at a global scale on a global directory.

Moreover, browsing can be general (all the services), or it can target a specific set of services (based on its type, domain, location, etc.). Depending on the granularity of the service descriptions and the service discovery protocol, a query may produce more or less specific answers. In practice, this means that the resolution phase may or may not be required (see Section IV for more details). Finally, a search engine can be utilized together with the discovery mechanism, in order to filter all the available services and resources that are of relevance to the clients.

III. Service Discovery Protocols for Low Power Networks

As already mentioned, the goal of service discovery protocols is to allow services and devices in the network to automatically become aware of each other without the need for human intervention[3]. Service discovery protocols may be broadly characterized by transport, complexity, topology and scope. Based on these characteristics, Figure 2 classifies the discovery protocols surveyed in this section. First, we describe service discovery protocols for traditional or non-constrained networks, in an effort to provide the readers a historical view of existing service discovery protocols and why they are not suited for constrained networks. Later, the survey focuses on service discovery protocols for traditional low power networks in which specifically developed protocols are used between constrained nodes, and gateways are required for protocol translations and where the focus is not to expose device functionality to the outer world. Finally, we review IP based protocols where remote direct end-to-end communications are enabled, thus facilitating a new range of applications where devices are not isolated any more but reachable by any other machine in the Internet.

The *transport* mechanism used by a discovery protocol usually depends on the application requirements. Most discovery protocols simply operate on top of the traditional IP layer using TCP or UDP as the transport protocol whereas others, such as Zigbee's discovery protocol, are intended for non-IP networks with specific resource constraints.

The *complexity* of a discovery protocol may be characterized by the power, computation and connectivity requirements of the protocol when searching for services and matching client queries. In order to keep the complexity low, search mechanisms must optimize the packet overhead incurred when querying for services. Moreover, query parsing should not require high power computation or memory resources. Some engines only allow searching on the basis of service type whereas others support more fine-grained search using complex queries. In the constrained networks, there must be a compromise between the query expression complexity and the computation overhead when designing search and query matching techniques.

[3] The term service used until Section IV generally refers to some functionality offered by a device, such as a printing or sensing service In Section IV onward, service will be technically defined for different protocols under consideration.

Additionally, the *scope* of the discovery protocol characterizes the set of resources that are discoverable by a given client. In the IP-based systems, the scope is usually limited to the resources discoverable within a subnet using multicast but it may as well be limited to the transmission range, for example in wireless personal area networks (WPANs) or infra-red systems. The scope also depends on the *topology* employed by the discovery system. Some systems may use one or more directories to manage resource registrations and client queries, whereas others may use a pure peer-to-peer or hybrid topology for service discovery. Therefore, large scopes may be achieved by using topologies that interlink multiple local directories.

A. *Discovery Protocols for Traditional Networks*

The choice of a discovery protocol depends on the intended applications. A wide range of discovery technologies are in use today for different purposes, the most popular ones being UPnP and Bonjour [23]. The UPnP stack, based on the simple service discovery protocol (SSDP) [24], is used by vendors such as Microsoft, Intel, Sony and Samsung in residential and office environments for discovering computers and digital entertainment appliances. SSDP uses HTTP notification announcements to discover services which are identified by a Uniform Resource Identifier (URI) and Unique Service Name (USN) containing resource type, and a URL pointing to the service description in XML. Because HTTP and XML are too expensive due to large overhead, UPnP is not well suited for service discovery in low power networks.

On the other hand, Bonjour is Apple's implementation of the IETF Zeroconf protocol mainly used in offices for the discovery of printers, computers and services. Bonjour discovery technology leverages the current Internet infrastructure through the combination of mDNS [21] with DNS-SD [14]. The low complexity of mDNS makes it a suitable candidate for low power networks and therefore it is currently being considered for service discovery in constrained networks by the IETF. The details of DNS-SD are explained in Section IV.

Although not as commonly used as UPnP and Bonjour, the Service Location Protocol (SLP) [3] is still supported by companies such as Hewllet-Packard, Novell and Oracle Solaris in their products to allow networking applications discover resources such as printers, fax machines and cameras. The SLP services are defined using URLs, and clients publish their existence through multicast service registration messages containing the service type, URL and attributes. In response, the clients reply with the URLs of matching services and their valid lifetime. In addition to direct discovery through multicasting, SLP supports the use of discovery agents (DA) which store service registration messages into a local data base. The main difference between SLP, UPnP and Bonjour is that SLP permits the use of complex search queries (based on the Lightweight Directory Access Protocol, or LDAPv3) which reflects its orientation towards enterprise service discovery. The drawback of these complex queries is high overhead per request which is unsuitable for low power networks.

In addition to the above most popular IP-based service discovery protocols, there also exist other protocols, like Universal Description, Discovery and Integration (UDDI) [25] for web services, Jini for Java objects, and Salutation [5]. However, these protocols are not suitable for constrained networks considered in this paper. In particular, UDDI is based on an XML message format which requires high processing power for data parsing. Jini and Salutation are based on remote service invocation methods in which the clients remotely invoke services through their available interfaces. This requires java remote method invocation or remote procedure calls (RPC) which are currently infeasible in constrained networks.

B. *Discovery Protocols for Constrained Networks*

As mentioned above, most of the discovery protocols for wired, powered or high-bandwidth wireless networks are not suitable for constrained networks. In order to meet the design requirements (see Section I), the general goal for M2M discovery protocols for constrained networks is that they must be resource efficient in terms of low processing and memory overhead, as well as must minimize the number and size of message exchanges.

Along this line, non-IP standard protocol specifications for wireless low power networks have defined their own resource aware discovery mechanisms. However, with increasing demand for IP connectivity down to the constrained devices, new service discovery protocols for constrained networks are being proposed based on the IP transport. Next, we detail both non-IP and IP based service discovery trends for constrained M2M communications.

Non-IP Based Service Discovery for Constrained M2M Communications

Over the last decade several wireless protocol specifications for constrained devices have defined their own mechanisms to facilitate device and service discovery. For instance, KNX-RF [26], Z-Wave [27] and EnOcean [28] are some of the well-known examples in home and building automation arenas. Although such communication protocols have recently gained attention due to their resource aware design, their discovery capabilities are very limited. This is because they usually depend on an operator performing some manual configurations, e.g. creating tables for binding devices or pushing buttons for device configurations.

The Bluetooth Service Discovery Protocol (SDP) [29] is another wireless low power protocol, but mainly used in mobile ad-hoc environments such as between a mobile phone and headsets or car stereo systems. Bluetooth device discovery is performed by periodically broadcasting and scanning for inquiries. Once the device address is known, the connection is established through pairing and only then the information about services available on devices is provided. Even though the latest Bluetooth specification (BLE or version 4.0) achieves low energy consumption and high pairing speed, SDP only works for Bluetooth devices which make it unsuitable for other types of low power devices.

Finally, the Zigbee device discovery provides the facility for devices to find node-wide (not application/endpoint specific) information about other devices in a network, such as addresses, manufacturer ID, types of applications running, power source, and sleep behavior. The device discovery is mostly used to learn about device capabilities, in particular for cases where a manufacturer implements extended ZigBee commands. In addition to the device related services, the Zigbee Device Profile (ZDP) also contains a variety of standard mechanisms, called ZDP service discovery, to identify the applications and their properties running on the devices. Service discovery is performed mainly during device configuration and integration into a ZigBee network by requesting a so-called simple descriptor from other devices of interest which describes everything to know about the endpoint: the Application Profile ID, supported input and output clusters.

Utilizing non-IP protocols as the transport mechanism has the drawback that service discovery is limited to the low power network domain. However, M2M systems require IP interoperability to discover services hosted on a back-end server network such as the Internet or among low power networks using different radio technologies. To enable low power networks based on the proprietary protocols to discover services in IP based network infrastructures, a gateway is required to convert the proprietary protocol used within the low power network to the protocols used in the TCP/IP domain. These gateways introduce single points of failure and usually depend on humans for static registration and management of resources.

For instance, it is possible in Zigbee to expose services provided by the sensor devices to an IP based network via a ZigBee IP Gateway [30] but it requires human intervention. The gateway configuration is accomplished by a two-phase process if the services provided by ZigBee devices are exposed via a REST interface. In the first phase, the ZigBee gateway needs to associate an endpoint with a local service descriptor provided by a user. In the second phase, the gateway needs to associate the response URI with the previously configured endpoint. As a result, messages sent to this endpoint by ZigBee devices will be forwarded to the specified URI. Once this procedure is finished, an application can send a REST request to the gateway with a URI including the ID of the preconfigured endpoint. The ZigBee network can respond to this request by sending a response to the same endpoint on the gateway. The described mechanism lacks flexibility as it requires human intervention to configure it. Moreover, there is no standard procedure to inform new IP applications about exposed ZigBee services in the gateway. Therefore, the service discovery mechanism is only limited to the ZigBee domain, and the exposed ZigBee services on the gateway can only be used by the application that configured the gateway accordingly.

IP Based Service Discovery for Constrained M2M Communications

As more and more M2M applications require the interconnection of low power networks to the Internet, both the research and industry communities have designed diverse IP based service discovery protocols for constrained networks. As early as 2005, Sensinode Ltd. [31] started developing a nano-IP stack including a reduced version of HTTP and a directory-less version of SLP that uses WAP Binary XML (WBXML) compression mechanisms for reduced parsing complexity. However, the nanoSLP protocol does not support the use of multicast and has very limited search capabilities. In the same year, the Simple Service Location Protocol (SSLP) for 6LoWPAN [32] also started its standardization in the IETF Networking Group. SSLP uses Tokenized XML strings to minimize packet exchanges, and adds support for translation agents (TA) to the standard SLP framework. The TAs run on gateways to perform the translation between SLP messages in an IP network and SSLP messages in a 6LoWPAN network. However, this solution involves complexity and incurs delay in translation, each time a message is translated to or from the SLP.

On the other hand, human readable names are not necessarily required for M2M communications. For this reason, nanoSD [33] proposes the mapping of XML service descriptions from nanoSLP into attribute-value pairs using a defined mapping tree structure in the gateway. The content of the tree-based database may be dynamic so as to download new service description templates from the Internet. The use of attribute-value pairs proposed by nanoSD decreases the parsing complexity and packet overhead in the low power network. However, nanoSD performs multicast and broadcast extensively, and requires each node to keep the service information of its neighbors. Moreover, the tree mapping approach proposed at the gateway introduces single points of failure. Similarly, the ITU-T E.164 NUmber Mapping (ENUM [30] based service discovery protocol uses a mapping to reduce the packet overhead in DNS. Clients make queries based on human readable strings and the proposed service discovery procedure converts these into integers following a similar approach to the ENUM telephony standard [35].

More recently, the IETF working group for Constrained RESTful Environments (CoRE) has started designing resource aware service discovery protocols based on CoAP as well as proposing the use of existing Internet protocols such as DNS-SD for the same purpose. These protocols, intended to become a standard, have the advantage of being light weight and IP based, which guarantees power efficient operation and interoperability. Moreover, they are based on widely accepted technologies such as REST (CoAP) or DNS which guarantees easy adoption. Therefore, these protocols have recently gained significant attention.

Although other standards for low power wireless networks, such as ZigBee, have their own application layer protocols and service discovery mechanisms, their current objective is to move towards IP at the end device for increased compatibility and integration with other systems [36]. This is also the case for standards in the building automation domain, for example BACnet [37]. It can therefore be expected that these standards may eventually adopt CoAP or DNS based service discovery protocols.

Given the current trend towards IP based service discovery protocols for constrained networks, the next sections describe in details these protocols and analyze their performance, advantages and disadvantages.

IV. CoAP and DNS Service Discovery

CoAP and DNS-SD mechanisms are currently being considered by the Internet community for service discovery in constrained M2M networks. If DNS-SD is utilized for service discovery, it is considered as a complementary protocol to CoAP. One of the reasons behind considering DNS for service discovery is that it would incur only limited overhead as far as memory resources are concerned. This is due to the fact that CoAP defines URI schemes for identifying or locating CoAP resources, where the CoAP URI scheme has the format: "coap:" "//" host [":"port] path-abempty ["?" query]. The host part can either be an IP address or a registered name. If the host is a registered name, then a name resolution service such as DNS is required to obtain the device address. Note however that the overhead of CoAP and DNS frames is different and a human readable name would not necessarily be required for M2M communications.

So far we have used the popular expression *service discovery* as a general term to refer to the discovery of functionality offered by a device. Nevertheless, in the context of IP based constrained networks, the discovery of functionality can be decoupled into two technically distinguishable parts: *service* and *resource* discovery [38]. For completeness, the definitions of *service* and *resource* for CoAP and DNS protocols are given below.

When utilizing CoAP, a *service* is defined by the tuple {protocol, host, port}, that is, a *service* represents the entry point of a CoAP server, e.g. *coap://[2001::11]:5683*. In addition, a *resource* in CoAP is any feature of an end-point that allows REST based interactions, that is, it can be acted upon with CoAP methods and identified by a URI. For example, a CoAP *resource* may be expressed as *coap://[2001::11]:5683/sensor/tempC*. Moreover, CoAP *resources* are generally categorized by *resource type* (*rt*) attributes, e.g., rt=tempC. CoAP *services* (end-points) and *resources* are discovered with CoAP *resource* discovery methods as detailed in subsection IV.A. On the other hand, a *service* in a constrained network utilizing DNS-SD is defined by the tuple {service subtype, type (protocol)}, e.g., *_tempC._sub._coap._udp*. The types and subtypes are usually defined by a Standards Development Organization (SDO). The service definition together with the domain, e.g., *_tempC._sub._coap._udp.example.com*, is utilized to discover instances of the services, e.g., *1234Sensor._tempC._sub._coap._udp.example.com*. The DNS *Service* instances can later be resolved to end-points (CoAP *services*) and URIs representing CoAP *resources* as detailed in subsection IV.B. Figure 3 summarizes these concepts.

A. *CoAP Based Resource Discovery*

CoAP is a RESTful Web transfer protocol currently under development for M2M communications in constrained networks. The CoAP specification defines a client/server model similar to HTTP where a CoAP end-point can typically act as both the server and client. Since CoAP is for constrained networks, it introduces low header overhead and reduced parsing complexity. Additional features include: (i) unicast and multicast support; (ii) acknowledged and unacknowledged transactions; (iii) four different request methods similar to those of HTTP: GET (retrieve resource representation), POST (process request), PUT (update/create resource) and DELETE (delete resource); (iv) and three types of response codes: 2.xx (success), 4.xx (client error), 5.xx (server error).

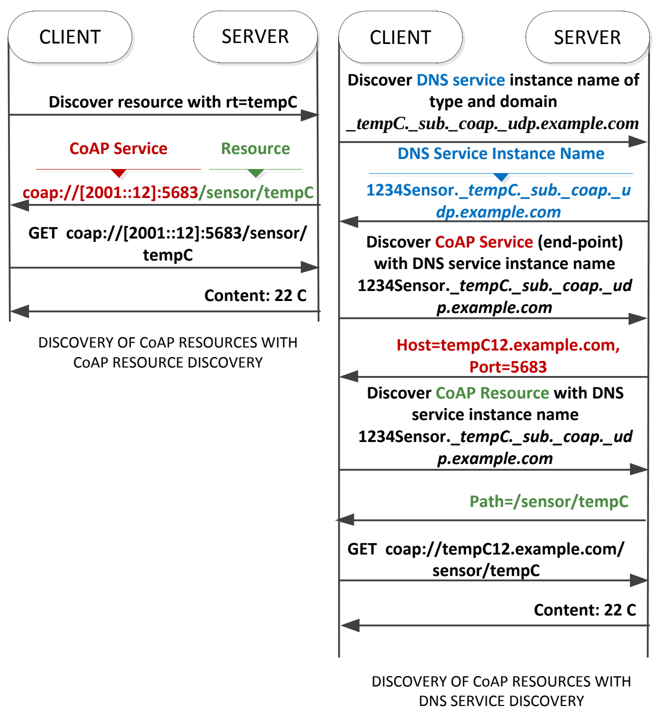
CLIENT
SERVER
Discover resource with rt=tempC
CoAP Service
Resource
coap://[2001::12]:5683/sensor/tempC
GET coap://[2001::12]:5683/sensor/tempC
Content: 22 C
DISCOVERY OF CoAP RESOURCES WITH CoAP RESOURCE DISCOVERY
CLIENT
SERVER
Discover DNS service instance name of type and domain _tempC._sub._coap._udp.example.com
DNS Service Instance Name
1234Sensor._tempC._sub._coap._udp.example.com
Discover CoAP Service (end-point) with DNS service instance name 1234Sensor._tempC._sub._coap._udp.example.com
Host=tempC12.example.com, Port=5683
Discover CoAP Resource with DNS service instance name 1234Sensor._tempC._sub._coap._udp.example.com
Path=/sensor/tempC
GET coap://tempC12.example.com/sensor/tempC
Content: 22 C
DISCOVERY OF CoAP RESOURCES WITH DNS SERVICE DISCOVERY

Figure 3: Definitions and High Level Usage Examples of Service and Resource Discovery Concepts for CoAP and DNS protocols.

With regard to resource discovery, two main mechanisms based on CoAP have been proposed: CoAP resource discovery and CoAP Resource Directory (RD). The main objective of these mechanisms is to supply URIs, also known as links, for the resources available within the server, together with attributes that describe those resources, and any possible link relation [19]. CoAP resource discovery is a distributed approach where a device discovers resources offered by another device by performing a direct query. In contrast, with CoAP RD, all resource discovery queries are performed on a centralized directory entity.

1) *Distributed CoAP Resource Discovery*

This is the basic method that may be utilized by a CoAP device to discover resources hosted by another device without the need for a directory. When a client device, for instance a light controller, needs to obtain the resources hosted by a server, such as a smart lamp, the server must issue a GET request to the well-known URI */.well-known/core* of the server as illustrated in Figure 4(a). If this request is a unicast because the target server address is known, then only the target server will respond with the URI of all its discoverable resources in the CoRE Link Format [19]. However, sending a multicast discovery request is also possible within a limited scope, if IP multicast is supported within the network, in order to discover the end-points (CoAP services) and their offered resources with a single query to the well-known URI.

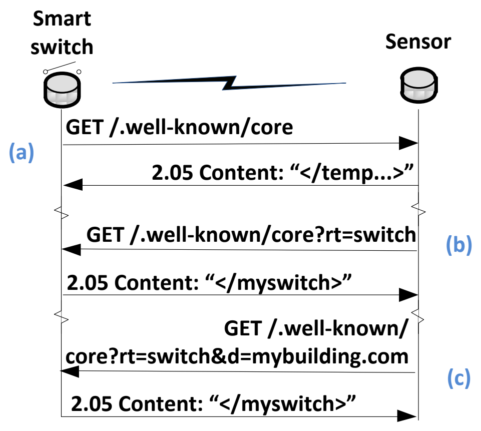
Smart switch
Sensor
(a)
GET /.well-known/core
2.05 Content: "</temp...>"
GET /.well-known/core?rt=switch
(b)
2.05 Content: "</myswitch>"
GET /.well-known/
core?rt=switch&d=mybuilding.com
(c)
2.05 Content: "</myswitch>"

Figure 4: Distributed Resource Discovery

Clients can also query for specific types of CoAP resources. This is achieved by utilizing a query string in the request method consisting of search parameters listed as *parameter=value* pairs. For instance, if a sensor requires to obtain only those resources of the type *switch*, a query string such as the one shown in Figure 4(b) could be used, where the parameter *rt* denotes the resource type. Moreover, several parameters can be used together in a query as shown in Figure 4(c), where a device looks for the resource descriptors corresponding to type *switch* within the *mybuilding.com* domain (denoted by parameter *d*). Additional link attributes that may be used to perform detailed queries include the interface description *if*, which relates types of resources to CoAP methods they accept [39], and the maximum expected size *sz* of a resource. Note that any server can decide which of its available resources to be discovered.

This distributed discovery method has the advantage that the queries are performed directly from the client to the server without requiring an intermediate directory. However, for this, a client needs to know the IP address or host name of the server that is queried, which means that either an external application would need to provide IP address/hostname or it would need to be hard-coded into the device's firmware. When the IP address or host name is not available, another possibility is to issue a multicast request. However, a multicast request is not reliable, i.e., the client does not have the means of knowing that the request has reached all intended destinations, and thus the client may not obtain the required information. Finally, if the IP address or host name is unknown, the client may issue a discovery request for each neighbor whose address may be obtained from the network layer, for instance, thereby obtaining their hosted resources in a reliable manner. However, serial unicast is not the most desirable method for a resource constrained network where communication must be kept to a minimum in order to preserve energy.

2) *Centralized CoAP Resource Discovery*

The CoRE working group proposes the use of a Resource Directory (RD) entity within LoWPAN to enable resource discovery [20]. Similar to any directory, the goal of an RD is to store descriptions of resources held by the servers (e.g. sensors, actuators) within the LoWPAN, and allow clients (e.g. other sensors, building management applications, etc.) to perform lookups on those resources. With this approach, all resources offered by the servers are registered automatically by them in this single centralized resource directory entity so that clients can discover any required resource with a single request. In order to use the RD either for registration or for performing a lookup, the device must first know how to reach the RD, i.e., its address. The end-point can locate the RD by a number of means: (i) assuming a default location such as the Border Router of the LoWPAN whose address is known by router advertisements; (ii) with the help of anycast address that may be assigned to the RDs; (iii) or by discovering the RD using the CoRE Link Format resource discovery as described in the last subsection is illustrated in Figure 5.

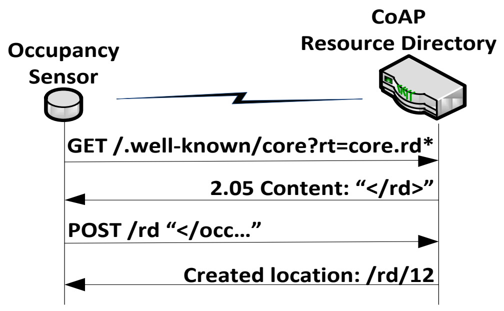

Figure 5: CoAP RD Registration Process

Once the RD is successfully discovered, a server can register its resources within the RD by performing a POST request to the path indicated by the RD in its response to the discovery request (i.e., */rd*). If the registration is successful, the RD returns the path to the end-point where its resources are located (i.e., */rd/12*). The RD may also proactively discover the resources of the different servers using the resource discovery method described in the next subsection.

Whenever a directory is used, additional protocol features must be available to manage the directory entries. For CoAP RD, the following interfaces are currently defined:

- **Update:** servers may update their registration if necessary, using the location path returned by the RD after registration and performing a PUT as illustrated in Figure 6(a).
- **Validation:** the RD may proactively check if any registration is valid by querying the end-point. Figure 6(b) shows the validation process where the *Etag* option represents the freshness or version of the resource being validated. If the *Etag* does not match the current representation, the most recent representation is sent to the RD enclosed in a successful CoAP *2.05 Content* response.
- **Removal**: registrations can be deleted by the end-points at any time by issuing a DELETE request to the registration path as depicted in Figure 6(c). The RD may also proceed to delete a registration when detecting that the life time of the registration has expired (where the life time of the resource is indicated by the end-point upon registration).
-

Once devices have registered their resources with the RD, they can start browsing the RD database to find any required resource. When a client wants to perform a lookup in the RD to find out which servers implement a certain resource, it has to issue a GET request to the RD. For this, the client utilizes a specific query to only obtain those results that match the client's interest. As an example, in Figure 7(a), a sensor device queries the RD to return those registered resources (i.e., lookup type *res*) whose resource type is switch (e.g. *?rt=switch*). In order to issue the query, the location of the directory should be known and obtained following the process as described in Figure 5. Because a query to the RD directly returns a CoAP resource, no resolution is necessary, that is, the requesting device can start interacting with the discovered resource immediately. Note that this applies to the distributed CoAP discovery method as well. Finally, with the help of CoAP RD, the CoAP services (end-points) or domains can also be discovered with the lookup types *ep (end point)* or *d (domain)* respectively (note that this is only defined for the directory based lookups and not available for the distributed resource discovery). Figure 7(b) shows how a sensor discovers the end-point (lookup type *ep*) that hosts a switch resource.

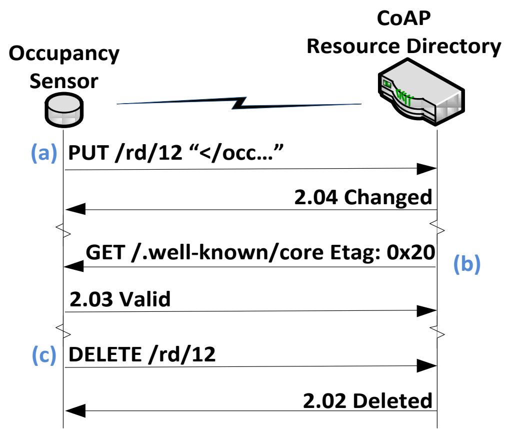

Figure 6: CoAP RD Entry Maintenance Interfaces

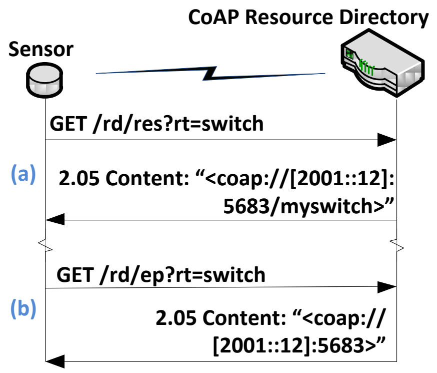
Sensor
CoAP Resource Directory
GET /rd/res?rt=switch
(a)
2.05 Content: "<coap://[2001::12]:5683/myswitch>"
GET /rd/ep?rt=switch
(b)
2.05 Content: "<coap://[2001::12]:5683>"

Figure 7: Sensor Node Performing CoAP RD Lookup

B. *DNS Based Service Discovery*

Domain Name System Service Discovery (DNS-SD) is a mechanism whereby standard DNS programming interfaces, packet formats, and servers can be employed to browse the network for services [14]. DNS-SD defines how to name and arrange the DNS records, namely pointer (PTR), service locator (SRV), IPv6 address (AAAA) and text (TXT), with the purpose of facilitating the service discovery within a subdomain. DNS-SD does not alter the structure of DNS messages, operation codes, record types or any other DNS protocol values.

Broadly speaking, a DNS-SD server contains a list of services that are defined in accordance with a Service Instance Name having format: *<Instance>.<ServiceType>.<Domain>*. For the specific case of LoWPANs for building automation applications, for example, it is suggested that the *<Domain>* part of the service definition should contain some location information (e.g. *room1.building2.example.com*) [38]. The *<ServiceType>* part (e.g., *light._sub._coap._udp*) follows some conventions that must be pre-established by Standard Development Organizations (SDO), such as the Organization for the Advancement of Structured Information Standards (OASIS) or IP for Smart Objects (IPSO) Alliance, which basically identifies an application protocol (service) and other additional functionality (service subtypes). Finally, the *<Instance>* part, e.g., *light1234*, identifies a specific instance of the service. This can be configured by the network operator/user, although a properly established default instance may allow the device or service to be accessed without requiring any manual configuration.

In order to relate services with instances, and instances with end-points that offer services, specific records are defined. Conceptually, a record can be viewed as a mapping between two parameters (e.g., service instance to IP address). More specifically, the records contain the following information:

- **PTR record**: maps *<ServiceType>.<Domain>* to a service instance name of that service *<Instance>.<ServiceType>.<Domain>*.
- **SRV record**: maps a service instance name to the service URI. This provides the basic description of a service in terms of hostname, port, priority, weight and lifetime or time to live (TTL).
- **AAAA record**: maps a service instance name to the IPv6 address.
- **TXT record**: maps a service instance name to different resources and attributes of a service, i.e., gives detailed information of a service by mapping the service instance to key=value pairs (e.g. path=/temp, if=BACnet, etc.) such as service types, location, interfaces, etc.

A high level overview of the DNS service discovery process was illustrated in Figure 3. As shown there, the client would query for the PTR record in order to obtain the service instance name; for obtaining host and port information the client would query for the SRV record; and finally, in order to obtain the CoAP resource, the client would query for the TXT record which contains the path to the resource.

Finally, there are three ways of performing DNS-SD: unicast, multicast and extended multicast. The main difference lies in the manner in which they perform the discovery. With unicast DNS-SD, a central server containing service descriptions is queried. With multicast DNS (mDNS) [21] or extended multicast DNS (xmDNS) [22], no central server is available, which means that queries have to be multicast locally or within the site's local scope, respectively, to discover the required information.

1) *Distributed DNS Service Discovery: mDNS*

Multicast DNS (*mDNS*) extends the current DNS specification to networks without infrastructure, where the devices query their neighborhood (local domain) through multicast instead of querying a DNS server through unicast. The mDNS adds to the DNS specification a .local domain, a well-known port and address for the multicast, and defines how to manage multiple answers to a single query.

In the distributed DNS based service discovery, the devices are able to publish information about the services and resources they offer using mDNS advertisements, which have the same format as standard DNS queries but are sent to the IPv6 multicast address *FF02::FB*. These advertisements may include service types and name (PTR records) with the domain name (local in case of mDNS, or any other in case of a global domain), the host name and port (SRV record), the address (AAAA record), and finally the extended description of the device (TXT records).

Assuming a building automation control scenario as in the previous section, a smart light would publish its services by including in its mDNS advertisements the PTR, SRV and TXT records defined in Tables 1 and 2. In this example, the PTR record defines the mapping from a service instance name, e.g., *BULB1_bc._bulb._sub._coap._udp.testbed.local* (abbreviated in the tables as *BULB1_bc* for simplicity) to a *<ServiceType>.<Domain>* tuple denoted as *_light._sub._coap._udp.testbed.local*. The service type here defines the protocol used (e.g., CoAP/UDP) and the subtype is the resource to be accessed, e.g. light. Multiple PTRs can be defined for the same service to enable different query formats (see Table 1).

Table 1: PTR Records Example

```
_bulb._sub._coap._udp.testbed.local PTR BULB1_bc
            _coap._udp.testbed.local PTR BULB1_bc
```

The SRV records for the same service instance *BULB1_bc* describe how to access the service (see Table 2). This includes the hostname URI (e.g., *light1.testbed.local*), the port (e.g., *5683*), the priority (where zero indicates maximum priority), and relative weight for records with the same priority. TXT records are always defined in conjunction with SRV in order to provide additional description of the service. In this scenario, the TXT record provides the path to access the resources that activate the light (path=*/lt/2/on*) and the resource type (*rt=ipso.lt.on*).

Table 2: SRV and TXT Records Example

```
BULB1_bc IN SRV 0 0 5683 light1.testbed.local
         IN TXT path=/lt/1/on
         IN TXT rt=ipso.lt.on
```

Once an interested client such as a smart switch has all the information contained in the records, it will simply use the obtained information to access the light resource.

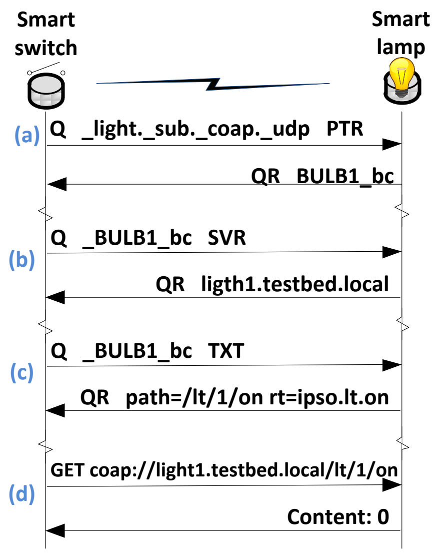

Figure 8: mDNS Browsing Example

The distributed DNS service discovery also allows the browsing of services hosted in other servers by means of sending standard DNS queries to the multicast address. This would allow the smart switch from our example to look for any lights in its local neighborhood (see Figure 8, where *Q* indicates Query and *QR* indicates Query Response). In this case, the DNS query would only include the PTR record with the service specific type that the switch wants to find, for example, *_light._sub._coap._udp*. Any device that matches the requested service, in this case a light using the CoAP protocol, will reply to the smart switch with the PTR record pointing to a service instance (e.g., *BULB1_bc*) shown in Figure 8 (a).

After receiving the instances that match the switch's interest, it has to resolve those to URIs that may be acted upon. For this, the switch asks for further records for those instances (SRV and TXT) which will provide IP address, port, path, and other relevant information as illustrated in Figure 8(b) and 8(c). Finally, the client device can use the obtained information to interact with the discovered functionality, in this case CoAP resources as shown in Figure 8(d).

2) *Centralized DNS Service Discovery: DNS-SD*

In the centralized DNS service discovery, a DNS-SD server is assumed to be available within the network infrastructure. The DNS-SD server stores service descriptions from devices in its subdomain. Similar to CoAP RD, the devices start by registering their services in the DNS-SD. However, in this case there is no standard DNS registration message. The most common implementation of the registration message, for instance such as performed with Bonjour, is to reuse the mDNS publication message to register service descriptions in the DNS-SD. This message may be sent by multicast following the same process described earlier, or may be sent directly to the unicast address of the DNS server if the address is known. Alternatively, the device may discover the DNS-SD server address through browsing the well-known service type *_b._dns-sd._udp.local*. Finally, when the DNS-SD server is in a different subnet, a different process, referred to as the global DNS-SD, must be used.

Currently no method is available with global DNS-SD to discover the address of a remote server location automatically. Thus, assuming that the DNS-SD server address is already known, for instance through IPv6 router advertisements, the smart lamp would register its service through a unicast publication message sent to the global DNS server. Since the remote DNS-SD server is not in the same subnet as the light, the server will not be able to observe the evolution of services and resources within that network, e.g., if the light is no longer reachable due to loss of connectivity. To solve this problem, the IETF has defined a synchronization mechanism referred to as dynamic DNS updates [40]. The Dynamic DNS updates utilize a parameter called lease life-time in the DNS-SD server records that is updated every *t* minutes (*t* = 30min is the suggested value). If the resource is not updated within this time frame, it will be removed from the DNS-SD register. Additionally, the IETF also defines DNS long-lived queries to allow the clients to observe any changes in the service registrations [41].

Finally, the DNS-SD can register all services automatically under the well-known service pointer *_services._dns-sd._udp._local*. This allows browsing all services registered in a directory, similar to the */.well-known/core* interface of CoAP.

Once the registration process ends, any device will be able to look up services through DNS queries to the DNS-SD server. The only difference with respect to the distributed case is that a single response from the DNS-SD server includes all the registered services in the network that match the requested type. After receiving the service instances, the resolution process must be carried out through the global DNS-SD server.

V. CoAP and DNS Service Discovery: Evaluation

This section evaluates CoAP and DNS based service discovery protocols in terms of several performance metrics: overhead, discovery functionality, interoperability, scalability, reliability, robustness, required human interaction and suitability for group communications which are presented below,

A. *Overhead*

Here we describe in detail a set of experiments that have been performed in order to analyze the overhead introduced by DNS and CoAP based service discovery protocols. The results include only the overhead introduced by CoAP and

DNS protocols, not considering UDP and IPv6 headers. For these experiments we utilized the simple scenario shown in Figure 9.

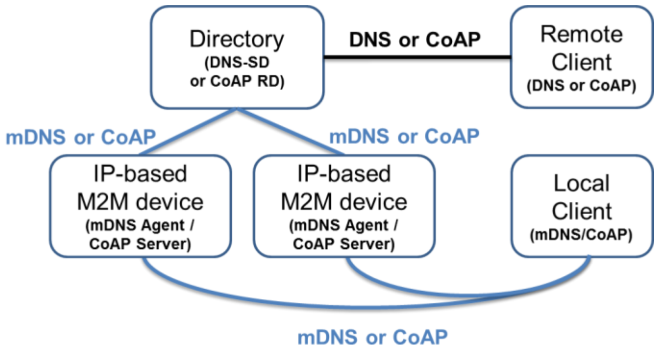

Figure 9: Experimental Network Setup

1) CoAP Ovehead Analysis

Figure 10 shows the overhead for the distributed and centralized CoAP resource discovery mechanisms for a star network scenario with two CoAP servers and one CoAP client as shown in Figure 9. For this evaluation, both servers implement a light service with *path=/lt/1/on* and resource type *rt=ipso.lt.on*. As observed from Figure 10, for the centralized case, the devices must discover and register their resources with the RD following the process described in Section IV.A (RD Discovery and RD Registration as per Figure 10). Since each device must discover and register with the RD in an independent manner, the overhead is directly proportional to the number of devices. The RD discovery process seems quite inefficient for a large number of devices, since the same RD discovery information has to be transmitted independently multiple times. The lookup overhead is also shown for the centralized case (RD Lookup as per Figure 10). As shown, the bulk of the overhead in this phase is associated with the response. This is because the RD must send the complete path to the requested resources for each registered device, for example *<coap://[2001::10]:5683/lt/1/on>;rt=ipso.lt.on*, *<coap://[2001::11]:5683/lt/1/on>;rt=ipso.lt.on*. Much of the information present in the URI could be omitted since only the address changes (e.g. from *2001::10* to *2001::11*); however the complete URI is sent by the RD. Thus, it would be interesting to investigate how this redundant overhead could be reduced.

On the other hand, for the case of distributed discovery, the nodes do not carry out the RD discovery and registration phases. Only the lookup is performed by an unreliable multicast (which in this case succeeds for all servers), or separate reliable serial unicasts (Multicast Lookup and Unicast Lookup as per Figure 10). In general, it can be seen that the distributed mechanism outperforms the centralized mechanism in terms of overhead as the RD discovery and registration phases do not have to be performed. Moreover, the distributed lookup is less expensive in terms of overhead since only the path to the resources is transmitted instead of the complete URI, e.g., *</lt/1/on>;rt="ipso.lt.on"*, as the IP address is inferred from the packet source address.

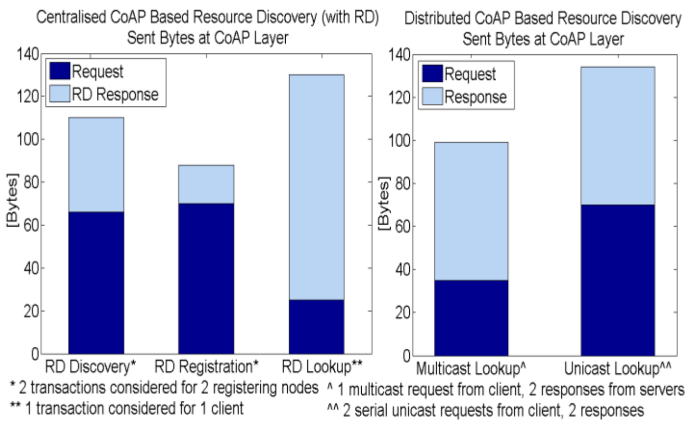

Figure 10: CoAP Based Service Discovery Overhead (2 Servers, 1 Client Star Network)

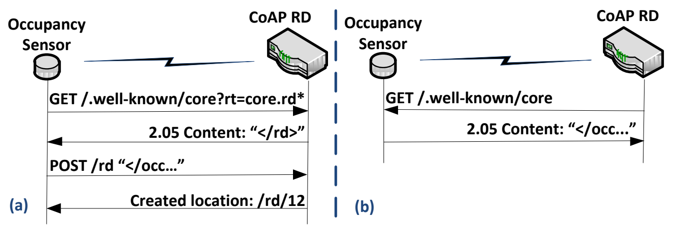

Figure 11: (a) Stateful and (b) Stateless Resource Registrations in the CoAP RD

Nevertheless, the obtained result for the distributed case has connotations: it has to be noted that a time-to-live (TTL) value of 1 was used for the multicast case, where the TTL represents the number of hops that the multicast packet can traverse. This means that the multicast request was sent once by the client and not forwarded by any of the servers. In the specific case of a star network, where the TTL can be easily established, a distributed service discovery introduces less overhead than using directories but it is less reliable. Note however that the lower overhead does not hold for a multi-hop network as the TTL must be set to a larger value. As long as TTL > 1, a single multicast packet will be forwarded, even if the request already traversed a node. Furthermore, note that a remote client located in a different site would not be able to perform multicast discovery. Thus, the multicast approach is more suitable for performing discovery in a small low power network, which does not require remote discovery. Alternatively, the RD may behave as a client utilizing distributed discovery to populate its tables (Stateless registration according to Section II.B). This would allow the use of distributed discovery while keeping a central directory for remote discovery which would only be proactively updated by the RD. Figure 11 shows examples of both stateful and stateless resource registration processes at the CoAP RD.

Because multicast requests are unreliable in the sense that some nodes may not overhear the request, the serial unicast distributed resource discovery case was also analyzed (see Figure 10). As can be observed from Figure 10, this discovery process is much more inefficient in terms of overhead than the multicast discovery. However, it is reliable. In addition, the serial unicast discovery requires the client to know the address of each of the available servers which in turn may require undesirable manual configuration.

Finally, for completeness, Figure 12 shows the same results for a large star network composed of 40 nodes. As can be seen, the same conclusions hold for the larger network, but more pronounced than in the smaller case. They include inefficient RD discovery process, redundant information transmitted in the RD lookup response, and inefficient serial unicast discovery.

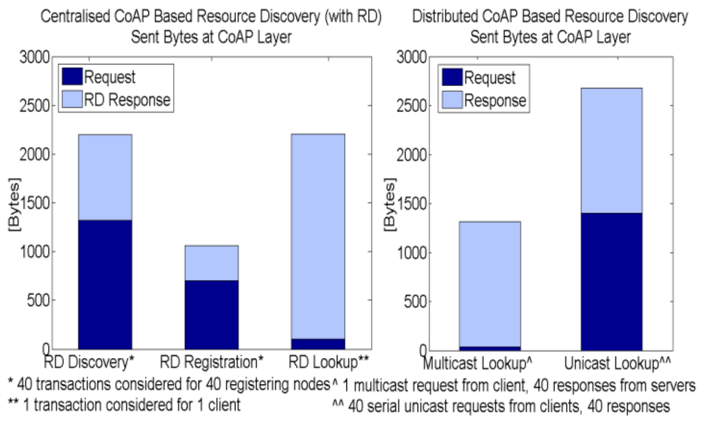

Figure 12: CoAP Based Service Discovery Overhead (40 servers, 1 Client Star Network)

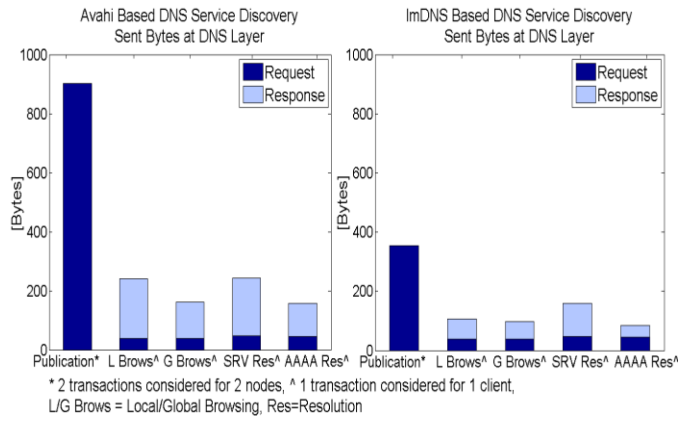

Figure 13: DNS-SD Overhead (2 Servers, 1 Client Star Network)

1) *DNS-SD Overhead Analysis*

This section analyzes the overhead associated with the DNS-SD protocol for service discovery. Again, the analysis is presented for different stages of the discovery process, i.e., publication, browsing (local and global), and resolution (local and global). Moreover, the results demonstate the overhead for two different DNS-SD implementations: Avahi [42] and light weight Multicast DNS (lmDNS) [43]. Avahi is a DNS-SD and mDNS free implementation for Linux-based Operating Systems (OS), while lmDNS is a lightweight implementation of mDNS for IP-enabled Smart Objects developed with the Contiki OS.

Figure 13 shows the overhead for the same scenario as before: a star network with two servers and one client. As can be seen, the Avahi implementation performs quite poorly in terms of overhead at the publication phase since it splits the publication in several messages. Initially up to three messages are sent to query other devices if they have selected the same service instance name in order to avoid name collisions (probing messages). Once collisions are discarded, the device can proceed to advertise the service (note that when collisions are detected the whole process has to be reinitiated for the new selected service instance name). For instance, for a light service, the following steps are followed in the performed experiments:

- **Service name resolution query (sent up to three times for probing):**
  mDNS Standard query **ANY**
  BULB_bc._coap._udp.local
  Authoritative records
  BULB_bc._coap._udp. **SRV** 0 0 *5683 light1.local*
  TXT "path=/lt/2/on"
  TXT "rt=ipso.lt.on"

- **Announcement of service:** Standard query response
  BULB_bc._coap._udp. **SRV** 0 0 *5683 light1.local*
  TXT "path=/lt/2/on"
  TXT "rt=ipso.lt.on"

  This includes the basic type with a pointer:
  _coap._udp.local **PTR** BULB_bc._coap._udp.local

  And also the additional pointers that we want to define (as per Table 1):
  _bulb._sub._coap._udp.local **PTR** BULB_bc._coap._udp.local

Note that the same information is transmitted four times: probing messages send the TXT and SVR records up to three times, and the advertisement sends TXT, SVR and PTR.

Probing is reduced with the lmDNS implementation to just one message and then advertisement is performed. Thus, if a name collision occurs, lmDNS has to take corrective measures.

As shown in Figure 13, the performance with lmDNS improves substantially at the publication phase as the probing messages are reduced. Therefore, this demonstrates the importance of how the discovery protocol is implemented for low power networks, since avoiding potentially unnecessary messages or control overhead can translate into significant energy savings.

With regard to browsing and resolution, lmDNS outperforms Avahi in terms of overhead again. This is due to the fact that Avahi includes all available records in the response whereas lmDNS only includes the information that is strictly necessary.

Finally, with regard to performing local or global (unicast) browsing, we can observe that there is no significant difference in terms of overhead for the lmDNS implementation.

For completeness, Figure 14 shows the results for a star network with 40 servers. We observe that the Avahi implementation is not suitable for low power networks due to the huge message overhead.

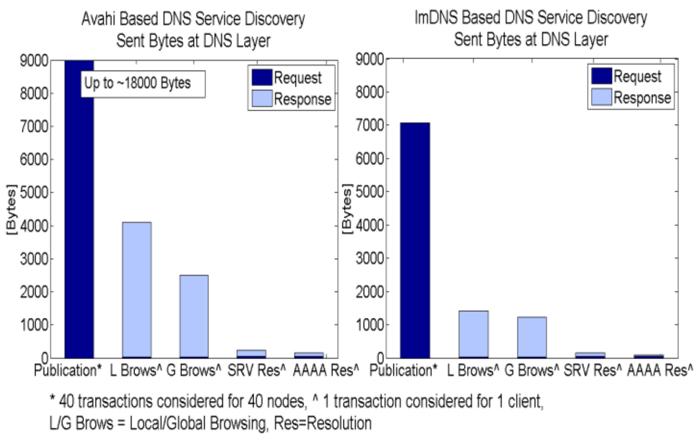

Figure 14: DNS-SD Overhead (40 Servers, 1 Client Star Network)

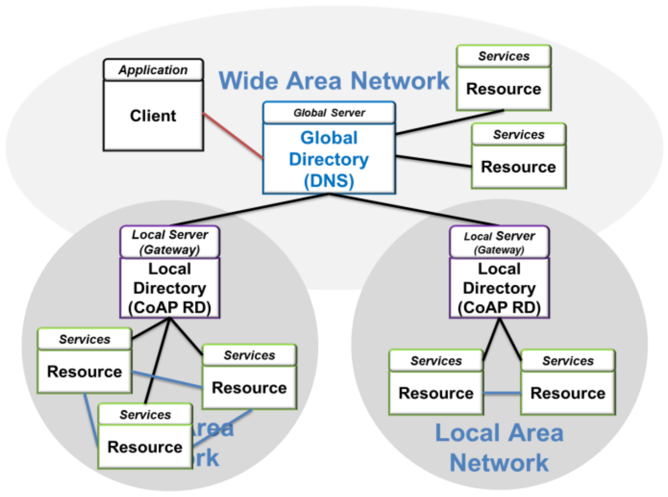

Figure 15: Architecture with Global and Local Directories

1) *DNS-SD vs. CoAP Resource Discovery*

Comparing the overhead incurred in service discovery by the lmDNS implementation to the distributed CoAP implementation, the results clearly demonstrate that CoAP is much more efficient. In the distributed case, only the multicast or unicast lookup phases are executed by CoAP with an overhead of around 100-150 bytes for the 2 server network case. On the other hand, for mDNS, during the advertisement, browsing, and resolution phases, message transmission incurs a total approximated overhead of 700 bytes. Thus, lmDNS introduces about seven times more overhead. Therefore, depending on how frequently the network changes, due to changes in connectivity, mobility, node deaths, etc., the higher overhead of lmDNS will have negative effects on the network in terms of energy consumption and lifetime.

In summary, CoAP discovery protocols show much better performance in terms of overhead than DNS discovery protocols. Indeed, CoAP is the most sensible approach when working with low power constrained devices. The DNS service discovery should only be considered while integrating with existing domain name systems as depicted in Figure 15. Similar to the binding proposed in [44], we suggest in those cases utilizing DNS-SD at the backend level and CoAP at the field level in order to reduce packet overhead.

B. *Discovery Functionality*

The IETF community considers the current options for service or resource discovery capabilities in IP based low power wireless embedded networks based on CoAP and DNS protocols, thereby providing use cases and recommendations for each option while developing these mechanisms further. Only time will tell if they will all coexist or one of them will finally prevail. Although these options are similar in concept, they provide different functionality in practice:

- CoAP allows discovery of different types of resources with one single request using queries of the format: *?key=value*. For instance, in order to find two types or resources, a query could be written as: *?rt=ipso.lt.on&rt=ipso.lt.dim* for a light switch and light dimmer resource. On the other hand, independent requests would have to be issued to perform the same type of discovery using DNS.
- CoAP allows the use of wildcard patterns such as: *?rt=core.rd** in order to get any resource that may start with *core.rd*. This is not supported by DNS.
- After a discovery request is issued, CoAP directly provides a resource that may be acted upon. On the other hand, DNS-SD provides service instances that later have to be resolved to IP addresses or host names.

To summarize, in general, CoAP based resource discovery allows a more efficient and richer set of mechanisms to perform lookups. Any device that implements CoAP is able to perform CoAP resource discovery, as no additional request methods are introduced. In addition, very little additional functionality is required for an end-point to register or perform lookups in an RD. On the other hand, DNS-SD is already employed as discovery mechanism for non-embedded devices, which means that using a DNS server for service discovery allows having a single centralized point where the discovery may be performed with a unicast DNS approach. If DNS is already required in a LoWPAN, then DNS-SD is easily implemented using standard DNS packet formats, queries, etc. Finally, mDNS requires additional memory in order to store service discovery information received from the neighborhood.

C. *Interoperability*

An important evaluation parameter for service discovery mechanisms is their interoperability. As mentioned earlier, some service discovery protocols for low power networks, such as those specified in the ZigBee standard, would only allow discovery between ZigBee enabled devices or between nodes and gateways. Thus, there is a clear lack of interoperability in their design. Since the protocols reviewed in this paper are based on DNS or RESTful Web technology, their interoperability with other IP based systems is assured. Given DNS service discovery is already employed widely, using this discovery method would likely allow almost immediate discovery by other existing systems. On the other hand, employing CoAP based discovery is also quite straight forward for those devices outside the low power network by installing additional protocol functionality in the legacy devices. It is worth mentioning that some initial proposals were made to combine both discovery systems by mapping CoAP RD registrations to DNS-SD records [44]. This allows employing CoAP at the low power network level and DNS-SD at the enterprise level (see Figure 15).

Finally, CoAP and DNS based service discovery protocols are independent of how the resource types or service instance names are defined. Thus, every SDO may define their own resource types and service instances for each of their devices as this will not influence the discovery performance. However, any device looking for a service defined by an SDO must know the standard name that the SDO has defined for the service. Therefore, the IPSO Alliance is currently defining standard resource types for CoAP based resources [45].

D. *Scalability*

Many M2M applications, such as building automation, require the deployment of a large number of devices to help monitor and control the performance of building equipment. For such applications, an important requirement is the scalability of the communication protocols used. As discussed, service discovery can be achieved in a distributed or centralized fashion. Although distributed approaches have the advantage that they do not require intermediate directories, they rely on sending multicast requests to perform discovery locally with distributed CoAP or mDNS or in a site with xmDNS. Thus, the scalability may be a problem for large scale multi-hop deployments. For instance, imagine a building floor where sensors and actuators are connected in a mesh network. Now consider that an occupancy sensor located in the main entrance needs to discover every light switch on the floor. When using xmDNS, the request would be flooded across multiple hops in the network as shown in Figure 16 (a). If every light switch is awake, up and running whenever this request is issued, the requesting sensor would receive an individual response from each of the switches. Depending on the devices' sleep schedules at the MAC layer, these responses would reach the sensor with different delays. As multicast is not reliable, some requests and responses may not be received along the multi-hop path which in turn may trigger further flooding of multicast requests. Along the same line, service advertisements are sent as multicast messages as well. Thus, a single device would potentially forward as many advertisements as there are nodes in the network, either once or multiple times, depending on how many advertisements are required. In summary, protocols based on the multicast are best suited for small networks, such as home area networks, or for those applications where discovery needs to be performed only in the local neighborhood (e.g., an occupancy sensor activating an appliance located one hop away). However, multicast protocols are not well suited for larger multi-hop networks where large numbers of flooded multicast packets would be too expensive in terms of energy and bandwidth.

With regard to the use of a directory, such as the CoAP RD, the scalability needs to be considered here as well. In a multi-hop network with RD location known (e.g., the RD is located in the 6LoWPAN border router), every node would have to issue a request to obtain the RD resource, e.g. *</rd>*, and forward any requests coming from its children in the routing path (this process corresponds to Figure 5 in Section IV.A.2). Similarly, every device will send a subscription request to the RD and forward those coming from its children, as illustrated in Figure 16(b). Once this is done, a node willing to discover a specific service offered by any node would issue a single unicast request to the directory which would respond with another single unicast request. Note that in order to maintain the freshness of the entries in the directory, the nodes will have to send unicast requests periodically to the directory; otherwise the directory will need to validate entries periodically.

Whenever a service fails, say due to battery depletion or connectivity problems, a node using a multicast based service discovery mechanism would likely have to send a new multicast request to find a similar service. On the other hand, when a directory is used, the node would send just a unicast request to the directory to find an alternative service. In conclusion, the use of a directory would be more appropriate for larger multi-hop networks as it provides better scalability than distributed service discovery mechanisms based on the multicast.

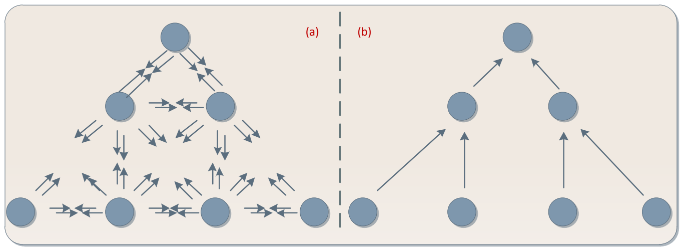

Figure 16: (a) Advertisement flooding vs. (b) Unicast Directory Registration

E. Reliability and Robustness

Reliability and robustness of the service discovery mechanism is a key to low power wireless networks where nodes may become unavailable due to battery depletion, bad channel conditions, mobility, or long sleep periods with the radio turned off.

Clearly, methods using multicast queries to advertise or discover services are prone to failures since multicast requests are unreliable. When a client sends a multicast request, it does not have the means to know whether the request has reached all the intended destinations. In other words, a device wanting to discover all the smart lights in a room may only end up discovering a subset, or even none of them. One possibility to counteract this problem is to send serial unicast requests or repeated multicast requests. Nevertheless, these solutions are expensive in terms of message overhead, which is undesirable in low power (constrained) networks. Given the limitations associated with the multicast, the IETF CoRE working group is currently considering the introduction of mechanisms to enable reliable multicast transactions [45].

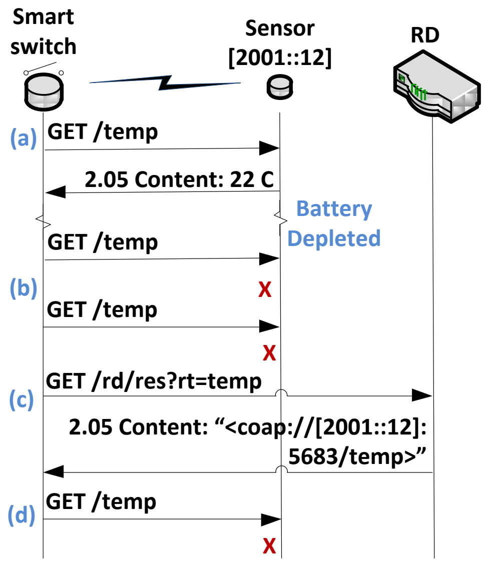

Figure 17: (a) Device Read Temperature Resource, (b) Attempt to Read Resource Fails, (c) Lookup to Outdated RD, (d) Persistent Failure Due to Outdated RD

To explain the robustness issues in a distributed service discovery mechanism, let us consider the failure of a node currently providing a service to a client. In such an event, the client will notice the failure as soon as it fails to receive information or acknowledgements from the device offering the service. Moreover, the client would need to find a similar service. For this purpose, the client checks its own database to see if a similar service was previously received through an advertisement, and the attempts to use it. Otherwise, the device attempts to directly send a multicast or unicast request to ask for a similar service. If the multicast or unicast request succeeds, the device starts using the new service immediately. In summary, a distributed service discovery mechanism can provide robust operation even when a service becomes unavailable as long as the discovery mechanism is reliable enough to provide alternative sources for a service.

On the other hand, with regards to the directory based service discovery, when entries of the directory are populated using reliable unicast requests, the devices can be sure that their services have been registered successfully. Thus, this method offers higher reliability than their multicast based counterparts. As regards to node failures, if a device has some other equivalent service details stored, it can attempt to use the alternative service directly. Otherwise the device has to look for a new equivalent service within the directory. For this purpose, the directory must also keep its entries as coherent as possible with the real status of the network so that the services which are no longer available are not discoverable. Therefore, the challenge here is to balance the need for sending frequent updates to keep the directories updated and reducing the energy consumption on control tasks to a minimum. Figure 17 shows an example of a resource failure case when utilizing an RD for resource discovery. After having discovered a temperature resource available in the node with address *[2001::12]*, the smart switch of the example starts reading the resource as in Figure 17(a). After a while the temperature sensor fails due to battery depletion. The switch becomes aware of this by failing to receive updates and decides to ask the RD for a new temperature resource (see Figure 17(b) and 17(c)). In this case, the RD is not coherent with the status of the network, that is, it still stores temperature resource in the node *[2001::12]* even though this is not reachable any more. Because of this unawareness, the RD returns the unavailable resource URI to the switch which in turn attempts to use the service again as depicted in Figure 17(d). To summarize, if we measure robustness as the capability of a device to find a new service when the current one fails, the robustness will directly depend on the ability of the system to keep the directory entries as coherent as possible with the real status of the network.

F. Human Interaction

One of the drivers behind introducing service discovery protocols into low power networks is the need to create independent, self-configurable systems. Service discovery mechanisms are a key element to achieving self-configuration at the application layer. The use of multicast DNS at the local level allows devices to discover each other locally. In contrast, the use of dynamic DNS updates allows devices to modify registrations in the global DNS servers in a unicast fashion [40]. Thus, when employing DNS, if service instances and host names are created autonomously by the individual devices, then the need for human configuration is eliminated. However, this human-free case may not be possible for some special M2M scenarios. For instance, in the building automation case, because location information needs to be embedded in service instances and host names, some manual configuration may be necessary [38] [47]. The same applies to CoAP when host names are used to identify the nodes instead of IP addresses. If manually introduced host names are not necessary, the CoAP resource discovery can work autonomously.

Finally, it is worth mentioning that the CoRE working group is currently defining standard service types for devices to find those servers providing application configuration information [45]. This will eliminate the need for a human operator to configure the operational parameters of individual devices, as this information would be obtained through the requested configuration service.

G. Groups

Another interesting service discovery feature useful for some M2M applications, such as building automation, is the ability to create and discover groups of similar services. For instance, when a single device such as an occupancy detector wants to use multiple similar services at the same time such as turning on several smart lights, it is useful to have groups of services.

For this purpose, while using unicast DNS, the DNS group records may be created manually by an operator and then registered to the DNS server. Alternatively, the DNS server may also create the records autonomously by grouping similar service types. Here, each group must be associated with a multicast address. A similar process can be followed for creating group resource records with CoAP.

It is to be noted that no mechanism has so far been proposed to autonomously create group service records for distributed CoAP resource discovery, mDNS, or xmDNS. Thus some additional functionality should be added to allow the creation of groups in a distributed fashion. Indeed, the IETF community is in the process of defining alternative mechanisms to perform CoAP based group communications, such as reliable multicast [45].

H. CoAP and DNS Service Discovery Enhancements for Low Power Networks

In the literature, there exist some solutions to improve the current CoAP resource discovery mechanisms proposed by the IETF CoRE Working Group. For example, TRENDY [48] aims at increasing the scalability of the CoAP RD solution by proposing a grouping mechanism based on the locations of devices. In order to reduce the traffic towards a single directory agent (DA) hosting an RD, TRENDY proposes the assignment of group leaders (GLs) in the sensor network. The GLs report to the DA and act as a mediator for the maintenance of resources belonging to their group members. In addition to reducing traffic towards the DA for status maintenance, the proposed approach allows the DA to execute location based commands. On the other hand, the authors in [49] improve the scalability of service discovery by using a hierarchy of linked CoAP servers and integrating them with DNS-SD, in order to discover sensors from any remote location through the Internet. In the sensor network, the devices are discovered from the gateway by sending CoAP GET requests to a well-known resource that allows the retrieval of its name and address. Then, the sensor gateway stores this information in a local DNS to act as a resolver of DNS requests coming from higher entities in the hierarchy, e.g., an Internet gateway.

There also exists literature focusing on the mDNS protocol in sensor networks. The authors in [43][50][51] have presented several lightweight implementations of mDNS with program code footprints ranging from 5KB to 10KB, which demonstrates their suitability for constrained networks. In order to facilitate lightweight implementations, the authors provide the following guidelines to reduce network traffic and processing: (i) do not respond to name and service requests directed to other nodes (additional section in the text entry); (ii) combine multiple TXT entries into single records; (iii) compress the text entries with data compression methods such as LZ-77 [52]; and (iv) use the IP layer buffer to generate DNS messages, thus reducing the processing required for handling DNS requests.

VI. Open Source Tools And Challenges

This section describes open source tools that may be used to experiment with both CoAP and DNS based discovery protocols. It also discusses some open issues and challenges.

A. DNS Service Discovery Open Source Tools

As surveyed in Section III.A, Apple's Bonjour is the most popular implementation for service discovery and is available for a variety of platforms, including Mac OS X, Windows, and Linux/BSD. Bonjour has been recently released under the Apache 2.0 license. Given the wide variety of available implementations, Bonjour is probably the best choice to test service discovery for enterprise or desktop applications.

The other option to test DNS-SD is Avahi, which was used for comparison in Section V of this paper. This implementation is available for Linux/BSD and multiple Linux-based embedded systems such as routers, perhaps the best choice for the development of gateway applications.

However, Bonjour and Avahi are still very heavy and thus difficult to integrate into resource constrained devices. For instance, Avahi makes use of Linux libraries, such as D-Bus and glib, which are not suitable for microcontrollers with limited processing capabilities. For evaluating and testing service discovery protocols in constrained devices, a lightweight version of mDNS agent should be implemented. The Contiki OS DNS *resolv* library can be used for this purpose [53].

B. CoAP Service Discovery Open Source Tools

In order to test CoAP based discovery algorithms several possibilities are available [54][55][56][57][58]. The most comprehensive implementations can be found for the Contiki OS [57] and for Java under the Californium project [58]. The Contiki OS version is suitable for low power devices whereas the Java implementation is more suitable for back-end applications. Both implementations are Open Source and have a community behind them to keep up with the latest standard developments.

C. Open Issues and Challenges

Several important open issues exist with regard to CoAP as it is still work in progress. For instance, for integration purposes, an initial approach was to allow the CoAP RD to populate the DNS-SD server records in an autonomous manner after performing mappings between CoAP resource descriptions and DNS service descriptions [44]. However, this work has currently been discontinued, most likely waiting for the RD proposal to be consolidated. Additionally, the current RD specification is still work in progress and does not provide details on how to register resources hosted by mirror servers in order to enable caching of resources[4]. Moreover, algorithms must be designed to balance control overhead and information freshness when utilizing directories in order to provide robust service discovery (see Section V.E). Furthermore, mechanisms to integrate legacy sensor networks with CoAP or DNS based systems should be available to allow the discovery of legacy services and devices.

With regards to security, one of the most critical challenges is to provide secure access to resources and secure registrations while respecting low power operation requirements [60][61]. This is particularly critical when the devices and their resources may be directly accessible over the Internet. Moreover, CoAP may be vulnerable to Denial of Service (DoS) attacks [19] when a request is multicast to the /.well-known/core interface.

VII. Conclusion

This paper presented a comprehensive overview of service discovery protocols for constrained M2M networks. As explained, although many service discovery protocols exist for traditional networks, they are not quite suitable for constrained networks where energy and computational efficiency is of paramount importance. Furthermore, some existing discovery protocols specifically designed for low power networks (e.g. ZigBee) have limited interoperability as they are just designed to work at the field level and require specific manual bindings at the gateway level. Due to these limitations, the IETF community is working on the specification of several discovery protocols for low power networks whose objective is to satisfy lower energy and time complexity requirements while maintaining high interoperability with other systems. These protocols, namely CoAP resource discovery and DNS-SD, offer different functionality with advantages and disadvantages as our performance evaluation results demonstrated. On one hand, CoAP provides higher granularity and higher efficiency in terms of lower overhead. On the other hand, DNS has the advantage of wide adoption in traditional networks, which has the potential to offer immediate integration. Because of such differing characteristics as well as dependency on the deployment scenario, it is not possible to predict at this stage which one of these two protocols will prevail.

[4] A mirror server is usually a line powered entity that contains copies of resources hosted by low power devices in order to allow fast access to sleeping device resources [59].

Acknowledgements: The authors are grateful to the anonymous referees for their insightful comments which helped us significantly improve the quality of the manuscript.

References

[1] M.R. Palattella, N. Accettura, X. Vilajosana, T. Watteyne, L.A. Grieco, G. Boggia, M. Dohler, "Standardized Protocol Stack For The Internet Of (Important) Things," IEEE Communications Surveys and Tutorials, DOI 10.1109/SURV.2012.111412.00158.

[2] F. Baker, D. Meyer. Internet Protocols for The Smart Grid. IETF RFC6272, June 2011.

[3] Bergmann, O.; Hillmann, K.T.; Gerdes, S., "A CoAP-gateway for smart homes," Computing, Networking and Communications (ICNC), 2012 International Conference on , vol., no., pp.446,450, Jan. 30 2012-Feb. 2 2012.

[4] M. Ersue, D. Romascanu, J. Schoenwaelder, Management of Networks with Constrained Devices: Problem Statement, Use Cases and Requirements. (IETF I-D work in progress), [Expired February 2013].

[5] W.K. Edwards. Discovery Systems in Ubiquitous Computing. IEEE Pervasive Computing, 5 (2) (2006), pp. 70–77

[6] http://www.upnp.org/ UPnP Forum, UPnP Device Architecture 1.1, October 2008.

[7] E. Guttman, C. Perkins, J. Veizades and M. Day. Service Location Protocol, Version 2. IETF RFC2165, June 1999.

[8] http://river.apache.org/doc/spec-index.html Jini Specification [Last accessed April 2013]

[9] http://salutation.org/wp-content/uploads/2012/05/originalwp.pdf Salutation Architecture Overview [Last accessed September 2012]

[10] N. Kushalnagar, G. Montenegro, C. Schumacher. IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. IETF RFC4919. 2007

[11] Z. Shelby, K. Hartke, C. Bormann, and B. Frank. Constrained Application Protocol (CoAP). (IETF I-D work in progress), [Expires October 2013].

[12] http://datatracker.ietf.org/wg/core/charter/ IETF CoRE Working Group [Last accessed April 2013]

[13] Z. Shelby, M. Garrison Stuber, D. Sturek, B. Frank and R. Kelsey. CoAP Requirements and Features. (IETF I-D work in progress), [Expired November 2011].

[14] S. Cheshire and M. Krochmal. DNS-Based Service Discovery. RFC 6763. ISSN: 2070-1721, Internet Engineering Task Force, February 2013.

[15] Zigbee Specification. ZigBee Document 053474r17, January 2008.

[16] C.N. Ververidis, G.C. Polyzos. Service Discovery for Mobile Ad Hoc Networks: a Survey of Issues and Techniques. Communications Surveys & Tutorials, IEEE , vol.10, no.3, pp.30-45, Third Quarter 2008

[17] A.N. Mian, R. Baldoni, R. Beraldi. A Survey of Service Discovery Protocols in Multihop Mobile Ad Hoc Networks. IEEE Pervasive Computing, vol.8, no.1, pp.66-74, Jan.-March 2009.

[18] P. Bellavista, A, Corradi, and C. Giannelli, Resource and Service Discovery. In *Mobile Agents in Networking and Distributed Computing* (eds: J. Cao and S. K. Das), Chapter 7, pp. 161-187, Wiley, July 2012 (ISBN: 978-0-471-75160-1).

[19] Z. Shelby, CoRE Link Format. Internet Engineering Task Force, RFC 6690.

[20] Z. Shelby and S. Krco. CoRE Resource Directory. (IETF I-D work in progress), [Expires August 2013].

[21] S. Cheshire and M. Krochmal. Multicast DNS. ISSN: 2010-1721, Internet Engineering Task Force, February 2013.

[22] K. Lynn and D. Sturek. Extended Multicast DNS. (IETF I-D work in progress), [Expires March 2013].

[23] Apple Bonjour https://developer.apple.com/bonjour/ [Last accessed April 2013].

[24] Y. Yaron, T.C. Goland, P. Leach, Y. Gu and S. Albright, Simple Service Discovery Protocol/1.0 Operating without an Arbiter, draft-cai-ssdp-v1-03 (IETF I-D work in progress), [Expires October 1999].

[25] Universal Description, Discovery and Integration http://uddi.xml.org/ [Last accessed April 2013].

[26] KNX Standard. System Specifications. Architecture. June 2009

[27] M. Galeev, Catching the Z-Wave. In technical papers and application notes on embedded & system design. EE Times India. October, 2006.

[28] EnOcean Radio Protocol. Specification v1.0. February, 2011.

[29] Bluetooth Service Discovery Application Profile (SDAP) https://www.bluetooth.org/Building/HowTechnologyWorks/ProfilesAndProtocols/SDAP.htm [Last accessed April 2013].

[30] Zigbee Alliance. Understanding ZigBee Gateway. How ZigBee extends an IP network. September 2010. https://docs.zigbee.org/zigbee-docs/dcn/09-5465.pdf [Last accessed April 2013]

[31] Sensinode Ltd. http://www.sensinode.com

[32] K. Kim, S. Yoo, H. Lee, S. Park, and J. Lee. Simple Service Location Protocol (SSLP) for 6LoWPAN. (IETF I-D work in progress), [Expires October 2009].

[33] A. Kovacevic, J. Ansari, and P. Mahonen. Nanosd: A Flexible Service Discovery Protocol for Dynamic and Heterogeneous Wireless Sensor Networks. International Conference on Mobile Ad-hoc and Sensor Networks, 2010.

[34] F. Anwar, M. Raza, S. Yoo, and K. Kim. ENUM Based Service Discovery Architecture for 6lowpan. In Wireless Communications and Networking Conference (WCNC), 2010 IEEE, pages 1-6. IEEE, 2010.

[35] Recommendation ITU-T E.164. Numbering Plan of the International Telephone Service. November 2011.

[36] Zigbee Alliance News: ZigBee Smart Energy Working Group Reaches Major Agreement on Use of HTTP and CoAP [Online] http://www.zigbee.org/, [Last accessed April 2013].

[37] BACnet IT Working Group, http://www.bacnet.org/WG/IT/index.html [Last accessed April 2013].

[38] P. van der Stok, K. Lynn, and A. Brandt. CoRE Discovery, Naming, and Addressing. (IETF I-D work in progress), [Expires January 2013].

[39] Z. Shelby and M. Vial. CoRE Interfaces. (IETF I-D work in progress), [Expires September 2013].

[40] P. Vixie, S. Thomson, Y. Rekhter and J. Bound. Dynamic Updates in the Domain Name System (DNS UPDATE), Internet Engineering Task Force, RFC 2136. 1997.

[41] S. Cheshire, M. Krochmal, K. Sekar. DNS Long-Lived Queries (IETF I-D work in progress), [Expires February 2007].

[42] Avahi implementation http://avahi.org/..[Last accessed April 2012].

[43] A. J. Jara, P. Martinez-Julia, A. Skarmeta. Light-Weight Multicast DNS and DNS-SD (lmDNS-SD): IPv6-Based Resource and Service Discovery for the Web of Things. Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012.

[44] K. Lynn, Z. Shelvy. CoRE Link-Format to DNS-Based Service Discovery Mapping. (IETF I-D work in progress), [Expires January 2012].

[45] Z. Shelby and C. Chauvenet. The IPSO Application Framework (draft-ipso-app-framework-04), August 2012.

[46] E. Dijk, A. Rahman. Miscellaneous CoAP Group Communication Topics. (IETF I-D work in progress), [Expires June 2013].

[47] Berta Carballido Villaverde, Julien Oury, Dirk Pesch, Rodolfo De Paz Alberola, Szymon Fedor. Demo Abstract: Commissioning of Low Power Embedded Devices with IPv6/CoAP, 10th ACM Conference on Embedded Networked Sensor Systems (SenSys), 2012.

[48] T. A. Butt, I. Phillips, L. Guan, G. Oikonomou. TRENDY: An Adaptive and Context-Aware Service Discovery Protocol for 6LoWPANs. In Proceedings of the International Workshop on the Web of Things (WoT), Newcastle, UK, June 2012

[49] I. Ishaq, J. Hoebeke, J. Rossey, E. De Poorter, I. Moerman and P. Demeester, Facilitating Sensor Deployment, Discovery and Resource Access Using Embedded Web Services. In Proceedings of the Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, 2012.

[50] Å. Östmark, J. Eliasson, P. Lindgren, A. van Halteren, and L.Meppelink, An Infrastructure for Service Oriented Sensor Networks. Journal of Computers, 1(5), pp. 20-29, 2006, doi:10.4304/jcp.1.5.20-29.

[51] R. Klauck and M. Kirsche. Bonjour Contiki: A Case Study of a DNS-Based Discovery Service for the Internet of Things. In Proceedings of the 11th International Conference on Ad-hoc, Mobile, and Wireless Networks (ADHOC-NOW'12), Springer-Verlag, Berlin, Heidelberg, pp. 316-329, 2012.

[52] J. Ziv and A. Lempel, A Universal Algorithm for Sequential Data Compression. IEEE Transactions on Information Theory, pp. 337-343, May 1977

[53] http://www.sics.se/~bg/telos/html/a00117.html Contiki Resolv Library. [Last Accessed April 2013].

[54] http://docs.tinyos.net/tinywiki/index.php TinyOS CoAP Implementation [Last Accessed April 2013]

[55] http://code.google.com/p/jcoap/ jCoAP Implementation [Last Accessed April 2013]

[56] http://coapy.sourceforge.net/ CoAPy Implementation [Last Accessed April 2013]

[57] http://people.inf.ethz.ch/mkovatsc/erbium.php Contiki CoAP Implementation [Last Accessed April 2013]

[58] http://people.inf.ethz.ch/mkovatsc/californium.php. Californium CoAP Framework [Last Accessed April 2013]

[59] M. Vial. CoRE Mirror Server. (IETF I-D work in progress), [Expires October 2013].

[60] Olaf Bergmann, Stefanie Gerdes, Silke Sch¨afer, Florian Junge, Carsten Bormann. Secure Bootstrapping of Nodes in a CoAP Network. WCNC 2012 Workshop on Internet of Things Enabling Technologies, Embracing Machine-to-Machine Communications and Beyond. 2012.

[61] B. Sarikaya, Y. Ohba, R. Moskowitz, Z. Cao, R. Cragie. Security Bootstrapping Solution for Resource-Constrained Device. (IETF I-D work in progress), [Expires January 2013].