Leveraging upon standards to build the Internet of Things

Jeroen Hoebeke, David Carels, Isam Ishaq, Girum Ketema,
Jen Rossey, Eli De Poorter, Ingrid Moerman, Piet Demeester
Department of Information Technology (INTEC)
Ghent University - iMinds
Ghent, Belgium
{firstname.lastname@intec.ugent.be}

***Abstract*— Smart embedded objects will become an important part of what is called the Internet of Things. However, the integration of embedded devices into the Internet introduces several challenges, since many of the existing Internet technologies and protocols were not designed for this class of devices. In the past few years, there were many efforts to enable the extension of Internet technologies to constrained devices. Initially, this resulted in proprietary protocols and architectures. Later, the integration of constrained devices into the Internet was embraced by IETF, moving towards standardized IP-based protocols. Long time, most efforts were focusing on the networking layer. More recently, the IETF CoRE working group started working on an embedded counterpart of HTTP, allowing the integration of constrained devices into existing service networks. In this paper, we will briefly review the history of integrating constrained devices into the Internet, with a prime focus on the IETF standardization work in the ROLL and CoRE working groups. This is further complemented with some research results that illustrate how these novel technologies can be extended or used to tackle other problems.**

Keywords—Internet of Things, constrained devices, IETF ROLL, IETF CoRE, RPL, CoAP, Web of Things, standardization

I. INTRODUCTION

Internet protocol technology is rapidly spreading to new domains where constrained embedded devices such as sensors and actuators play a prominent role. This expansion of the Internet is comparable in scale to the spread of the Internet in the ’90s and the resulting Internet is now commonly referred to as the Internet of Things (IoT). The integration of embedded devices into the Internet introduces several new challenges, since many of the existing Internet technologies and protocols were not designed for this class of devices. These embedded devices are typically designed for low cost and power consumption and thus have very limited power, memory, and processing resources and are often disabled for long-times (sleep periods) to save energy. The networks formed by these embedded devices are also constrained and have different characteristics than those typical in today’s Internet. These constrained networks have high packet loss, low throughput, frequent topology changes and small useful payload sizes.

In the past few years, there were many efforts to enable the extension of the Internet technologies to constrained devices, moving away from proprietary architectures and protocols. Most of these efforts were focusing on the networking layer: IPv6 over Low-Power Wireless Personal Area Networks (RFC4919) [1], Transmission of IPv6 Packets over IEEE 802.15.4 Networks (RFC4944) [2], IETF routing over low-power and lossy networks [3] or the ZigBee adoption of IPv6 [4]. These standardization efforts enable the realization of an Internet of Things, where end-to-end connectivity with tiny objects such as sensors and actuators becomes possible.

However, not global connectivity was at the basis of the great success of the current Internet, but web service technology. Today, an embedded counterpart of web service technology is needed in order to exploit all great opportunities offered by the Internet of Things and turn it into a Web of Things. Recently, standardization work has started within the IETF Constrained RESTful Environments (CoRE) working group [5] to allow precisely the integration of constrained devices with the Internet at the service level.

With these technologies, it has now become possible to deploy a sensor network, to interconnect it with IPv6 Internet and to build applications that interact with these networks using embedded web service technology. In this paper, we will briefly review the history of integrating constrained devices into the Internet, with a prime focus on the IETF standardization work in general and embedded web service technology in particular. This review is complemented with some research results that illustrate how these novel technologies can be extended or used to tackle other problems.

The remainder of this paper is organized as follows. In Section II, we discuss the evolution from proprietary solutions towards IP-based integration of constrained devices. In sections III and IV we will discuss IETF standardization work on RPL and CoAP respectively, complemented with own research results where applicable. In section V, we conclude and outline some future work and opportunities.

II. INTEGRATION OF CONSTRAINED DEVICES INTO THE INTERNET

In the absence of widely accepted standard protocols for resource-constrained devices, many vendors were encouraged to develop proprietary protocols to run inside their sensor networks. Connectivity between the Internet and the sensor

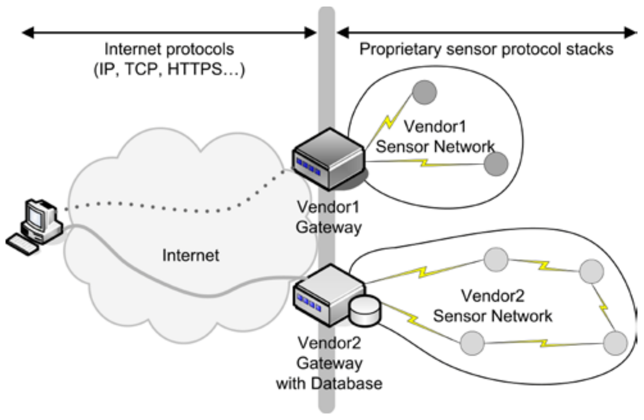


Figure 1: Gateways and properietary protocols are often used to interconnect sensor networks to the Internet

networks was achieved through the use of gateways or proxies. These gateways have to translate between protocols used in the Internet and proprietary protocols used in the sensor networks. Figure 1 displays two various sensor networks that are connected to the Internet by gateways. Users on the Internet have to connect to the gateways in order to obtain data from the respective sensor network. There are several ways how a gateway can handle such user requests. For example, the gateway from vendor 1 translates standard Internet protocols into proprietary sensor protocols and relays the requests to the sensors in its network. It then gets the answers from the relevant sensors by means of the proprietary sensor protocols and sends back the appropriate replies to the user using standard Internet protocols. The gateway offers an API that applications should use in order to create requests that can be understood by the gateway. Alternatively, the gateway of vendor 2 contains a database with pre-collected sensor data. When it gets a request from a user on the Internet, it replies directly to the requester using the data in the database. In some cases, the gateway is simply running a web server that makes the data available to the outside world. The user usually cannot tell, whether the returned data is coming in real-time from the sensors or whether it is coming from a value that has been previously stored in a database.

It is clear that such an approach does not enable a real integration of sensors into the Internet. Due to the lack of real end-to-end connectivity or interaction with the resource-constrained devices, flexibility of usage is reduced. Users can query the sensors only in the way that is allowed by the gateway. Adding new sensor resource often requires adaptation on the gateway. Another disadvantage is the vendor lock-in. Gateways and sensors often have to be from the same vendor in order to be compatible.

This raised the need for standardized solutions for network communication with constrained devices in tandem with the need for interoperability with the most widely used protocols in the Internet, initially IP and, in a later stage, HTTP. To address these needs, the IETF has formed several working groups: IPv6 over Low Power WPAN (6LoWPAN) [2], Routing Over Low Power and Lossy Networks (ROLL) [3] and Constrained Restful Environments (CORE) [4]. The 6LoWPAN group tackles the transmission of IPv6 packets over IEEE 802.15.4 networks, the ROLL group develops IPv6 routing solutions for low power and lossy networks and the CoRE group aims at providing a framework for resource-oriented applications intended to run on constrained IP networks. Together, these protocols allow the IP-based integration of constrained devices into the Internet in a standardized way, as shown in Figure 2.

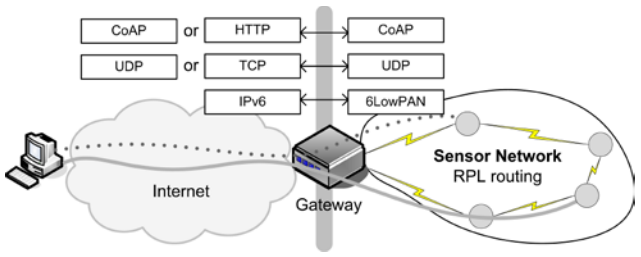


Figure 2: Internet protocols are extended to the sensor networks. The Gateway translates between the two protocol stacks.

Similar to the previous category of approaches, gateways are still used to translate between protocols used in the Internet and protocols used in the sensor networks, e.g. IPv6 to 6LoWPAN and vice versa. However the difference here is that through the use of standard protocols, many of the disadvantages from the previous approaches are now taken care of. For example it is now possible to have the gateway and the sensors from different vendors. Flexibility is also improved by this approach. Users can now query the sensors without the need for the gateway to understand the query and the data itself. The application payload can now travel directly from the client to the sensor, where it is processed and acted upon. The gateway takes care of the translation between standardized protocols. This makes adding and removing sensor resources transparent to the gateway and improves interoperability of devices.

In the following sections, we will briefly discuss in more detail the key concepts and protocols resulting from the ROLL and CoRE groups, complemented with own research results where applicable.

III. IETF ROLL AND RPL

Constrained networks, also called Low power and Lossy Networks (LLNs) have, due to their specific characteristics, specific requirements that are not satisfied by existing routing protocols such as AODV, OLSR, OSPF, etc. The ROLL working [6] group focuses on building routing solutions for LLNs. More specific, the working group focuses on industrial, connected home, building and urban sensor networks for which different routing requirements were specified. Also routing security and manageability issues and transport characteristics are kept in mind during protocol design. One of the realizations of this working group is the design of the IPv6 Routing Protocol for LLNs, also called RPL.

A. RPL concepts

RPL [7] was designed with the objective to meet the routing requirements for the different application scenarios for LLNs. The protocol provides a mechanism whereby multipoint-to-point, point-to-multipoint and point-to-point traffic are supported. Although RPL was specified according to the routing requirements for LLNs, its use is in no way limited to these applications. RPL routes are optimized for traffic to or from one or more roots that act as sinks for the topology. As a result, RPL organizes a topology as a Directed Acyclic Graph (DAG) that is partitioned into one or more Destination Oriented DAGs (DODAGs), with one DODAG per sink. A DODAG is constructed based on the rank information of the nodes. The rank represents the individual position, relative to other nodes with respect to a DODAG root. Rank information is derived from the analysis of the received DODAG Information Object (DIO) messages of neighboring nodes. To discover the neighborhood, a node may use DODAG Information Solicitation (DIS) messages to solicit a DODAG Information Object from a RPL node. Downward routes (root to leaf) are constructed using Destination Advertisement Object (DAO) messages. RPL supports two modes of Downward traffic: Storing (fully stateful) or Non-Storing (fully source routed). In the Non-Storing case, the packet will travel all the way to a DODAG root before traveling down. In the Storing case, the packet may be directed down towards the destination by a common ancestor of the source and the destination prior to reaching a DODAG root.

B. Mobility support for RPL

Currently the RPL protocol does not support collection of information from mobile nodes, missing functionality we added to RPL. When optimizing the RPL protocol to support mobile nodes, the interoperability with the existing protocol is important. Therefore the behavior of the current protocol, for fixed nodes, is preserved as much as possible and only the use of existing techniques is allowed. By adapting the objective function for mobile nodes the selection of a parent can be influenced, to select faster and a better appropriate parent for the moving node. The switching of parent is influenced by the fading of the link quality due to the movement of the node. This implies that the importance, of the new measurements of link quality, increase in comparison to the importance of the history (ETX percentage) of the link quality. However the importance of new link quality measurements should not be overestimated, because the network may be unstable.

To always have up-to-date link state information of the parent of a mobile node, active monitoring is required. The sending of a DIS message will force the parent to provide the mobile node with recent information. By the selection of an alternative parent for a mobile node, and also monitor it actively, an alternative link will be available when the connectivity with the preferred parent is lost. A DIS message (broadcast) can also be used to monitor the environment for new neighbors when moving across the network. Due to the limited resources of LLNs energy efficiency and minimal overhead has a high importance. Therefore in LLNs a balance between energy efficiency and reliability has to be made. It is obvious intensive monitoring of neighbors will increase reliability, but it will also increase the energy consumption.

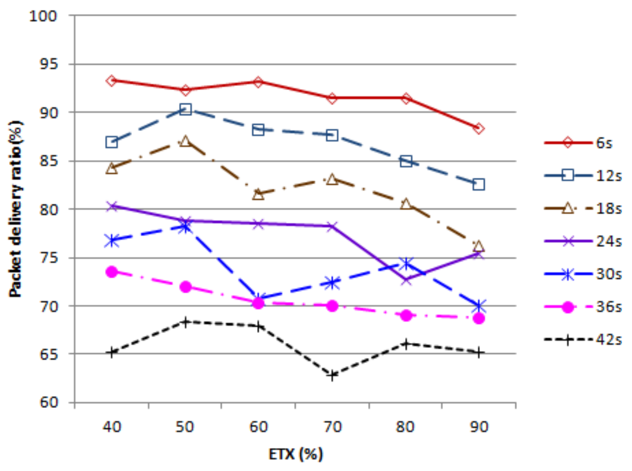


Figure 3: Packet delivery ratio for different percentages of history link estimation and DIS broadcast intervals

We implemented the proposed solution in the Contiki OS and validated its feasibility through simulations for different speeds (5 until 20 km/h) and different parameter combinations. The simulations indicate that reliability of the collection of information from a mobile node is improved. In Figure 3, the increase in packet delivery ratio is represented when increasing the monitor frequency of parents and environment. Also the negative effect of increase in importance of history link estimation is illustrated.

IV. IETF CoRE and CoAP

More recently, in 2010, an IETF working group, called Constrained RESTful Environments (CoRE), was founded specifically to work on the standardization of a framework for resource-oriented applications, allowing realization of RESTful embedded web services in a similar way as traditional web services, but suitable for the most constrained nodes and networks. Their work resulted in the Constrained Application Protocol (CoAP), a specialized RESTful web transfer protocol for use with constrained networks and nodes [8].

A. CoAP concepts

CoAP uses the same RESTful principles as HTTP, but it is much lighter so that it can be run on constrained devices [9-10]. As a result, CoAP has a much lower header overhead and parsing complexity than HTTP. It uses a 4-bytes base binary header that may be followed by compact binary options and a payload. Optional reliability is supported within CoAP itself by using a simple stop-and-wait reliability mechanism upon request. Secure communication is also supported through the optional use of Datagram Transport Layer Security (DTLS). The CoAP interaction model is similar to the client/server model of HTTP. A client can send a CoAP request, requesting an action specified by a method code (GET, PUT, POST or DELETE) on a resource (identified by a URI) on a server. The CoAP server processes the request and sends back a response containing a response code and payload. Unlike HTTP, CoAP deals with these interchanges asynchronously over a datagram-oriented transport such as UDP and thus it also supports multicast CoAP requests. This allows CoAP to be used for point-to-multipoint interactions which are commonly required in automation.

The IETF CoRE working group considers the constrained restful environments as an extension of the current web architecture. The group envisions that CoAP will complement HTTP and that CoAP will be used not only between constrained devices and between servers and devices in the constrained environment, but also between servers and devices across the Internet [11]. An important requirement of the CoRE working group is to ensure a simple mapping between HTTP and CoAP so that the protocols can be proxied transparently. Thus proxies and/or gateways play a central role in the constrained environments architecture. These proxies have to be able to communicate between the Internet protocol stack and the constrained environments protocol stack and to translate between them as needed.

Resource discovery is important for M2M interactions, and is supported in CoAP using the CoRE Link Format [12]. A well-known URI "/.well-known/core" is defined as a default entry-point for requesting the list of links about resources hosted by a server. Once the list of available resources is obtained from the server, the client can send further requests to obtain the value of a certain resource. The example in Figure 2 shows a client requesting the list of the available resources on the server (GET /.well-known/core). The returned list shows that the server has, amongst others, a resource called /s/t that would return back the temperature in degrees Celsius. The client then requests the value of this resource (GET /s/t) and gets a reply back from the server (23.5C).

In the following sections, we give two examples on how we use CoAP in our research. The first example is a useful extension to the CoAP protocol to realize a lightweight publish-subscribe mechanism embedded in the protocol. The second example shows how CoAP can be used to facilitate the discovery and deployment of sensors.

B. *Conditional observations using CoAP*

In addition to the main CoAP draft, a number of extensions have been proposed. One of those extensions is the observation of resource states through the introduction of the observe option, which allows clients to register with servers to be notified whenever the state of a resource changes [13]. A client interested in observing a resource includes the option in its GET request. Whenever there is a change of the resource state, the server sends a notification to the client. As such, observe offers the possibility for a client to have an up-to-date representation of the resource without the client having to constantly poll for changes. If the client acts upon these states and is only interested in specific states, it is up to the client to filter out the values sent by the server, discarding resource states that are not significant enough for its purpose.

A better alternative to these observations in combination with client-side filtering could be to specify filtering criteria when sending the observe request. To avoid this, we proposed a new CoAP option "Condition" as an extension to the Observe Option in order to support conditional observations [14]. This option can be used by a CoAP client to specify the conditions the client is interested in. Several condition types, i.e. filtering options, have been identified based on realistic use cases. Using conditional observations, the CoAP server will send a notification response with the latest state change only when the criterion is met. Using this mechanism a client can indicate that it is interested only in temperature values above 25C and not in all state changes.

This mechanism has been implemented on Erbium, a low-power REST engine for Contiki [15] and evaluated in Cooja, using Sky nodes having an MSP430 16-but CPU running at 3.9MHz and having only 48kB of ROM and 10kB or RAM. As such the feasibility of implementing this on a constrained device is proven. Further, we evaluated the correct operation for a simple scenario and showed that the use of conditional observations can result in a reduced number of packets and power consumption compared to normal observe in combination with client-side filtering. For instance, Figure 5 shows the per node average power consumption for normal observe (filtering at the client) and for conditional observe. Three different thresholds for the condition "All Values Greater Than" were used and the input temperature values were 100 pseudo-random integers between 20 and 29 with an average value of 24. The result clearly shows that a lightweight

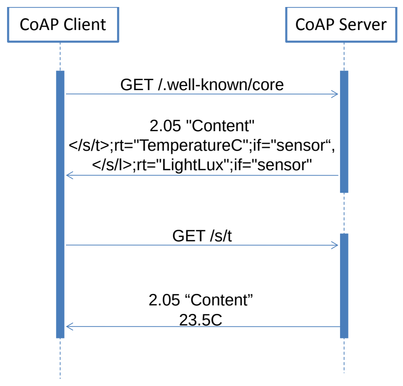


Figure 4: An example of CoRE resource discovery and CoAP request

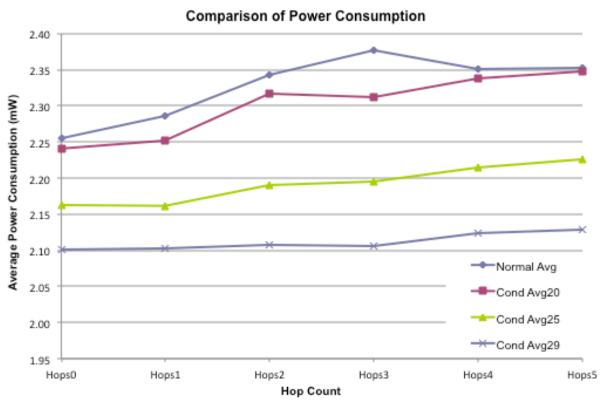


Figure 5: Per-node average power consumption for data collection using normal observe and client-side filtering and using conditional observations (3 different thresholds)

filtering mechanism on the constrained device can reduce the power consumption. Of course, the concrete gain will depend on the conditions of interest and thus the actual use cases: more extreme conditions will lead to larger gains. By realizing this functionality as an extension of the CoAP protocol, it can be seen as a resource independent enabler that realizes frequently used application logic. For more details about conditional observations we refer to [16].

C. Facilitating discovery and deployment using CoAP

With the presented IETF protocols, it has become possible to deploy a sensor network, to interconnect it with IPv6 Internet and to build applications that interact with these networks using embedded web service technology. Within the sensor network itself, the available protocols are largely self-organizing, requiring no human intervention. Also, if the IPv6 address of a sensor is known, its resources can be accessed using CoAP. Nevertheless, there are still several important hurdles that need to be overcome. Several gaps exist with regard to the automatic discovery of sensors, integration with current Internet standards such as DNS, user-friendly access to sensors from within a web browser or the fact that several manual configuration steps are still needed to integrate a sensor network within an existing networking environment. However, the advent of open standards for embedded web services on e.g. sensors and sensor gateways, offers new opportunities to tackle several of these challenges related to the deployment of sensor networks and the realization of global user-friendly connectivity and access to sensor resources by making use of embedded web services through the CoAP protocol.

Based on this observation, we implemented a novel self-configuration and bootstrapping mechanism in order to facilitate the deployment of sensor networks and enable the discovery, end-to-end connectivity and service usage of newly deployed sensor nodes. The proposed approach makes use of CoAP and combines it with DNS in order to enable the use of user-friendly fully qualified domain names (FQDN) for addressing sensor nodes. It includes the automatic discovery of sensors and sensor gateways and the translation of HTTP to CoAP, thus making the sensor resources globally discoverable and accessible from any Internet-connected client using either IPv6 addresses or DNS names both via HTTP or CoAP. As such, the proposed approach provides a feasible and flexible solution to achieve hierarchical self-organization with a minimum of pre-configuration. It bridges the gap between the deployment of constrained objects and the actual consumption of their services by users, services or other machines.

The overall process is summarized in Figure 6 and described in detail in [17]. Initially every newly deployed sensor knows its short address (network prefix not yet known) and a name (e.g. hardware ID) and this information is available via a well-known CoAP resource. The sensor gateway can now use a multicast CoAP request to query this resource on all sensors (pull-based) or the sensors can anycast or unicast this information to the gateway (push-based). Upon reception of this information, the gateway will store the information, create a complete IPv6 address (using the sensor subnet prefix) and Fully Qualified Domain Name (using the domain assigned to the sensor network). This information is then used to dynamically update the local DNS running at the sensor gateway (note that the sensor gateway acts as resolver of DNS requests for names in the sensor domain). When a sensor is no longer available (it doesn't reply to the periodic broadcasts), the information is removed from the local DNS. The same discovery process can be repeated at a higher level in the network hierarchy assuming that the sensor gateways also run CoAP servers. In case the Internet gateway notices that the sensor gateway does not have a subnet prefix, domain suffix and name configured, the Internet gateway can take this information from a pool of subnets and domains and send it as a CoAP POST request to the sensor gateway, which will update its configuration accordingly. By applying this mechanism and creating a hierarchy of linked CoAP servers,

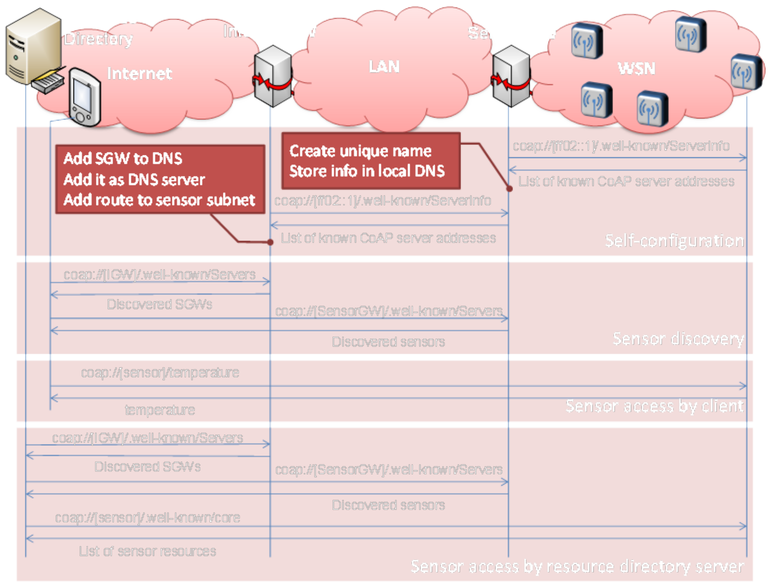
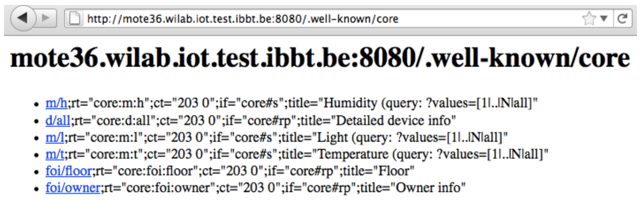

Directory
Internet
LAN
WSN
Add SGW to DNS
Add it as DNS server
Add route to sensor subnet
Create unique name
Store info in local DNS
coap://[ff02::1]/.well-known/ServerInfo
List of known CoAP server addresses
coap://[ff02::1]/.well-known/ServerInfo
List of known CoAP server addresses
Self-configuration
coap://[IGW]/.well-known/Servers
Discovered SGWs
coap://[SensorGW]/.well-known/Servers
Discovered sensors
Sensor discovery
coap://[sensor]/temperature
temperature
Sensor access by client
coap://[IGW]/.well-known/Servers
Discovered SGWs
coap://[SensorGW]/.well-known/Servers
Discovered sensors
coap://[sensor]/.well-known/core
List of sensor resources
Sensor access by resource directory server

Figure 6: Complete self-organization process, sensor discovery and resource access by a client and by a resource directory server



```
http://mote36.wilab.iot.test.ibbt.be:8080/.well-known/core

mote36.wilab.iot.test.ibbt.be:8080/.well-known/core

• m/h;rt="core:m:h";ct="203 0";if="core#s";title="Humidity (query: ?values=[1|..|N|all]"
• d/all;rt="core:d:all";ct="203 0";if="core#rp";title="Detailed device info"
• m/l;rt="core:m:l";ct="203 0";if="core#s";title="Light (query: ?values=[1|..|N|all]"
• m/t;rt="core:m:t";ct="203 0";if="core#s";title="Temperature (query: ?values=[1|..|N|all]"
• foi/floor;rt="core:foi:floor";ct="203 0";if="core#rp";title="Floor"
• foi/owner;rt="core:foi:owner";ct="203 0";if="core#rp";title="Owner info"
```

Figure 7: Example of accessing /.well-known/core on a sensor node using its FQDN name and going via a transparent HTTP-CoAP proxy.

any client (a human, another machine or a resource directory server) can easily discover and use any sensor without a lot of network overhead. Figure 6 shows how this can be used both by a client and a resource directory, which maintains an overview of all resources.

To enable HTTP access in our solution, the sensor gateway and the Internet gateway were extended in such a way to not only act as CoAP servers, but also as HTTP-CoAP proxies capable of translating HTTP messages to CoAP messages and vice versa. Clients can access these gateways via their favorite web browser using HTTP requests. The gateways map the requests to CoAP and send the requests to the sensors. Once the sensor replies using CoAP, the reply is sent back to the client using HTTP. A transparent mode is foreseen , where a client can directly use the IPv6 address or FQDN of the sensor and the TCP connection is intercepted and translated to CoAP. In addition to the mapping between HTTP and CoAP, the proxy implementation on the gateway also performs automatic rewriting of response in the CoRE Link format into HTML, so that it can be interpreted directly by the web browser and easily understood by humans. This solution has been implemented on the gateways using our modular C++ CoAP framework and on the sensors using an available CoAP implementation, it has been deployed on a publicly reachable testbed and evaluated on our experimental facility w.iLab.t. Figure 7 gives an example of how a client accesses /.well-known/core on a sensor using HTTP and FQDN names from a web browser, resulting in a web page with links to all resources.

V. CONCLUSION

In this paper, we gave a high-level overview of IETF standardization work for realizing the Internet of Things. Standardized or almost standardized protocols enable the integration of constrained devices in the IPv6 Internet, both at the network level and at the service level. The IETF groups 6LoWPAN and ROLL are focusing on the network connectivity and interoperability, whereas IETF CoRE focuses on realizing an embedded counterpart for RESTful web services. Anyone involved in Internet of Things research, whether dealing with network layer aspects or service layer aspects will, sooner or later, be confronted with these protocols. This could encompass simply using these protocols to realize IoT services, studying extensions and enhancements to these protocols or leveraging upon these protocols to solve open issues in the IoT world. In this paper, we have briefly illustrated how we enhanced RPL to support mobility, extended CoAP to achieve a lightweight publish-subscribe mechanism and build upon embedded web service technology to facilitate the deployment, discovery and resource access to IoT objects. It shows that the advent of standardized protocols is not an end point, but only a starting point for exploring many more of the open issues in realizing the IoT such as resource representations, security, dealing with sleeping nodes, energy efficiency, integration with existing web service technologies and tools, linking with Cloud services, use of semantics, easy creation of applications, scalability, interoperability with other wireless standards etc. . This paper merely touches the surface of this broad domain and aims encouraging others to further explore the world of Internet-connected objects and tackle other open issues and challenges.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n°258885 (SPITFIRE project), from the iMinds ICON projects GreenWeCan and O'CareCloudS, and a VLIR PhD scholarship to Isam Ishaq.

REFERENCES

[1] N. Kushalnagar, G. Montenegro, and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks: Overview, Assumptions, Problem Statement, and Goals", IETF RFC 4919, Aug. 2007.

[2] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", IETF RFC 4944, Sept. 2007.

[3] Routing Over Low power and Lossy networks (roll) http://datatracker.ietf.org/wg/roll/

[4] ZigBee Alliance Plans Further Integration of Internet Protocol Standards, https://docs.zigbee.org/zigbee-docs/dcn/09-5003.pdf

[5] Constrained RESTful Environments (core) http://datatracker.ietf.org/wg/core/

[6] Charter of the Routing Over Low power and Lossy networks (roll), IETF charter-ietf-roll-03, August 2009.

[7] T. Winter et al., "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", IETF RFC 6550, March 2012.

[8] Z. Shelby, K. Hartke, C. Bormann and B. Frank, "Constrained Application Protocol (CoAP)", draft-ietf-core-coap-12, work in progress, October 2012.

[9] D. Yazar and A. Dunkels, "Efficient Application Integration in IP-Based Sensor Networks", Proc. First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings, 2009.

[10] W. Colitti, K. Steenhaut and N. De Caro, "Integrating Wireless Sensor Networks with the Web," Proc. workshop on Extending the Internet to Low power and Lossy Networks, 2011.

[11] Z. Shelby, "Embedded Web Services", IEEE Wireless Communications, pp. 52-57, Dec. 2010.

[12] Z. Shelby, "CoRE Link Format", draft-ietf-core-link-format, IETF RFC 6690, August 2012.

[13] K. Hartke, "Observing Resources in CoAP", draft-ietf-core-observe-07, work in progress, October 2012.

[14] S.T. Li, J. Hoebeke and A. J. Jara, "Conditional observe in CoAP", draft-li-core-conditional-observe-03, work in progress, October 2012.

[15] M. Kovatsch, S. Duquennoy and A. Dunkels, "A Low-Power CoAP for Contiki", Proc. of the 8th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2011), pp. 855-860, 2011.

[16] G. Ketema, J. Hoebeke, I. Moerman, P. Demeester, S.T. Li and A. J. Jara, "Efficiently observing Internet of Things Resources", Proc. of The IEEE International Conference on Cyber, Physical and Social Computing, November 2012, to appear.

[17] 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", Proc. of the Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, pp. 717–724, October 2012.