Facilitating sensor deployment, discovery and resource access using embedded web services

Isam Ishaq, Jeroen Hoebeke, Jen Rossey, Eli De Poorter, Ingrid Moerman, Piet Demeester
Department of Information Technology (INTEC)
Ghent University – IBBT
Ghent, Belgium
{isam.ishaq, jeroen.hoebeke, jen.rossey, eli.depoorter, ingrid.moerman, piet.demeester}@intec.ugent.be

Abstract—**Smart embedded objects such as sensors and actuators will become an important part of the Internet of Things. With recent technologies, it has now become possible to deploy a sensor network and interconnect it with IPv6 Internet. However, several manual configuration steps are still needed to integrate a sensor network within an existing networking environment. In this paper we describe a novel self-organization solution to facilitate the deployment of sensor networks and enable the discovery, end-to-end connectivity and service usage of these newly deployed sensor nodes. The proposed approach makes use of embedded web service technology, i.e. the IETF Constrained Application Protocol (CoAP). Automatic hierarchical discovery of CoAP servers is one of the key features, resulting in a browsable hierarchy of CoAP servers, up to the level of the sensor resources, which can be accessed both over CoAP and HTTP and through the use of either DNS names or IPv6 addresses. To demonstrate the feasibility of our approach we have implemented the solution and deployed it on a test setup, which is publicly accessible to everyone.**

Keywords-CoAP; self-organization; Internet of Things; DNS; proxy; embedded web services; discovery

I. INTRODUCTION

The ubiquitous 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 new challenges, since many of the existing Internet technologies and protocols were not designed for this class of devices. These devices are typically optimized for low cost and power consumption and thus have very limited power, memory, and processing resources and have long sleep periods. 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. 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 network (ROLL) [3] or the Zig-Bee adoption of IPv6 [4]. Only recently, work has started to allow integration at service level. The IETF Constrained RESTful Environments (CoRE) working group is in the process of realizing the Representational State Transfer (REST) architecture in a suitable form for the most constrained nodes and networks. To that end the Constrained Application Protocol (CoAP) was introduced, a specialized RESTful web transfer protocol for use with constrained networks and nodes [5]. CoAP realizes a subset of REST that is common with the Hypertext Transfer Protocol (HTTP), but is optimized for machine-to-machine (M2M) applications.

With these technologies, it has now become possible to deploy a sensor network and interconnect it with IPv6 Internet. 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 several gaps related 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.

In this paper, we will describe novel self-configuration and bootstrapping mechanisms 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.

Following this introduction we give an overview of the challenges facing sensor network deployment, discovery and access (Section II) and the upcoming solutions for the realization of embedded web services (Section III). In sections IV and V, we then introduce a solution based on CoAP and DNS for the hierarchical self-configuration of sensors and show how it can be accessed via HTTP. Our implementation and first test deployment is then described in Section VI. Finally we point to related works and compare them with ours and conclude in Sections VII and VIII.

II. CHALLENGES IN SENSOR NETWORK DEPLOYMENT

The deployment of sensor networks, including their integration in the Internet, is a multi-faceted problem. First of all, there is the deployment of the sensor network itself, starting with the provisioning of the hardware, followed by the actual installation and optimal placement of the sensing infrastructure [6]. Once installed and activated, it is up to the communication protocols to create a fully operational sensor network that is robust, energy efficient and capable of delivering the sensor data to the sinks or sensor gateways. Looking at the literature and standardization bodies (see Section I), it is clear that there have been many efforts to create such protocols, including MAC layer protocols, addressing, routing, data collection protocols, etc [7]. As such, sensor network self-configuration, i.e. the creation of an operational sensor network and communication inside the sensor network can be considered as a well-studied problem for which several solutions exist.

The next aspect is connectivity with the IP-based Internet. On the one hand, sensor networks using proprietary networking solutions can be integrated with the Internet by deploying appropriate gateways that can do the translation from and to the sensor network. On the other hand, there is significant momentum for IP-based sensors and actuators as illustrated by the IETF work mentioned in Section I and several research papers. As such, the feasibility of integrating sensor networks with the Internet and enabling IP-based connectivity to sensors has been shown and made possible, for example in [8-11].

However, this is only the starting point. Next to the connectivity within the sensor network and the connectivity with the Internet, there are many other aspects related to the deployment of sensor networks. When a sensor subnet is connected to the Internet it needs to receive an address prefix, routing to the sensor network should be configured, ideally it should integrate with current Internet standards such as DNS, etc. Typically, manual interventions are still needed by an administrator. In addition, connectivity can be achieved, but knowing which sensors are present, discovering them and being able to use them in a user-friendly way that does not require any technical skills (e.g. from a web browser) is an interesting challenge that has only begun to receive more attention from the research community recently. It is clear that there are still many open aspects and challenges. In this paper, we describe a novel solution that is capable of dealing with several of these challenges. To this end, we have taken a fresh approach, making use of embedded web services.

III. THE ADVENT OF EMBEDDED WEB SERVICES

Recent research on embedded web services is laying the ground for a better integration of sensor resources into the service web. Since the dominating web protocol HTTP is too complex, the IETC CoRE working group, formed in 2010, has designed a simpler web protocol - CoAP. It uses the same RESTful principles as HTTP, but it is much lighter so that it can be run on constrained devices [12-13]. As a result, CoAP has a much lower header overhead and parsing complexity than HTTP. It uses a 4-bytes binary header that may be followed by compact binary options and a payload. Optional reliability is supported within CoAP itself.

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) 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.

Resource discovery is important for M2M interactions, and is supported in CoAP using the CoRE Link Format [14]. 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 Fig. 1 shows a client requesting the list of the available resources on the server (GET /.well-known/core). The returned list showed that the server has, amongst others, a resource called /sensors/temp that would return back the temperature in degrees Celsius. The client then requested the

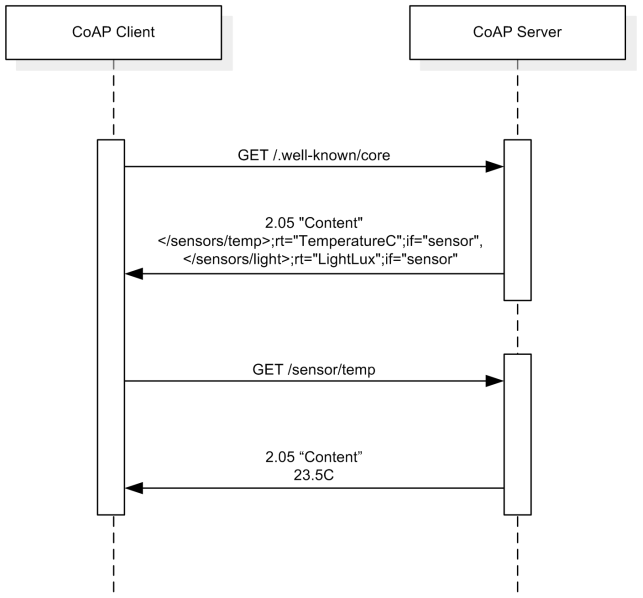


Figure 1. An example of the CoRE resource discovery

value of this resource (GET /sensors/temp) and got a reply back from the server (23.5C).

IV. Hierarchical self-configuration solution based on CoAP

By making use of the functionalities offered by CoAP, we have designed a hierarchical self-configuration solution that facilitates the deployment, discovery and resource access for sensor networks. In this section, we will present our approach in more detail.

A. *Assumptions*

In order to be able to design a self-configuration solution, one always has to make a few assumptions about certain aspects that have been preconfigured already. For example, in order for a new PC to auto-configure its globally routable IPv6 address, a router advertisement daemon has to be active in the network announcing the prefix of the network. Of course, the challenge is to restrict the required amount of pre-configuration involving humans as much as possible and to avoid these configuration steps at deployment time of the devices that dynamically join the network.

1) The network that will be extended with sensors

For our self-configuration solution, we assume the following basic network topology. An organization is connected via an Internet gateway, which also acts as DNS server, to the IPv6 Internet and has obtained a /48 IPv6 range and suffix for its domain names (e.g. "test.ibbt.be"). From this /48 range, a network administrator can assign subnets to different networks. For example, a /64 subnet is assigned to the LAN network behind the Internet GW (e.g. "iot.test.ibbt.be"). Now assume the organization wants to equip its building with sensors, which will be connected to the LAN network via one or more sensor gateways. The administrator reserves a pool of /64 subnets, domain name suffixes and sensor gateway names that can be assigned to newly deployed sensor networks.

2) The sensor network

Sensors should be considered as dumb devices that only have a minimal knowledge. For our self-configuration solution, we make the following assumptions. The sensor knows or will discover – via an address assignment protocol - its address in the sensor network. Typically, because of the limitations of a sensor device, these addresses are preferably small, e.g. only 16-bit for 6LoWPAN short addresses [15]. The complete IPv6 address of the sensor is not known, since it also depends on the sensor network the node will be deployed in. In the remainder of the paper, we will assume the use of 16-bit addresses for the sensors. Further, we assume the sensor knows its (or a) name. This name could be anything and could be for example a hardware identifier. A user-friendly name such as temperature_room1 would require user intervention and knowledge about the location where the sensor will be deployed. This can be done after deployment, where the automatically generated name can be replaced by a more meaningful name. Finally, the sensor runs a minimal CoAP server. Since the proposed solution makes use of CoAP, this is a strict requirement. Further, this minimal CoAP server should offer a well-known resource (/.well-known/serverInfo) that allows the retrieval of its name and address. No assumptions are made about the protocols inside the sensor network. These can be standardized or proprietary and it is the responsibility of the sensor gateway to translate from the IPv6 world to the sensor world. For the sensor gateway we also assume it will run a CoAP client and server and that it knows its global IPv6 address in the LAN.

B. *The solution*

Based on the assumptions in Section A, we will now describe how newly deployed sensors can be discovered and made accessible to the outside world.

1) Sensor discovery

After deployment of the sensors, the sensor gateway can be triggered to send a multicast (ff02::1) CoAP GET request for the resource /.well-known/serverInfo as shown in Fig. 2. By doing so, the sensor gateway will be able to discover the (short) address and name of all sensor nodes present in the network. The sensor gateway could also be configured to periodically perform the multicast. This allows verifying whether already discovered sensors have disappeared. Of course, in this case the multicast frequency should be limited in order to limit the resulting energy consumption. As an alternative, which we are currently investigating, it is also possible to let the sensor nodes announce their presence to the gateway, once or periodically (push versus pull-based solution).

Fig. 3 shows the manual execution and the corresponding result of such a multicast CoAP request into the sensor network (using the usb-to-sensor network interface). The figure shows that sensor3 and sensor4 are present in the network, next to some other sensors (not visible in the figure). If the sensor gateway has obtained a subnet prefix and domain for the sensor network (see next paragraph), it can construct the complete IPv6 address and FQDN of the sensor. 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 request for names in the sensor domain). As such, the sensor gateway has a list of all available sensors, which can be updated dynamically. When using periodic multicasting and when a sensor is no longer available, the information is removed from the local DNS. It is important to note that these updates to the DNS are restricted to the sensor gateway and stay within the administrative domain of the company.

2) Sensor gateway discovery

The same process can be repeated at a higher level in the network hierarchy assuming that the sensor gateway also runs a CoAP server. The Internet gateway can periodically

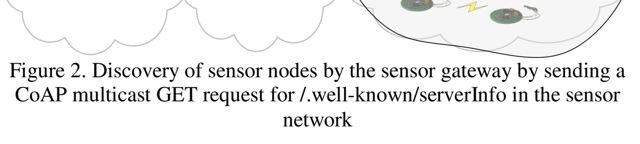
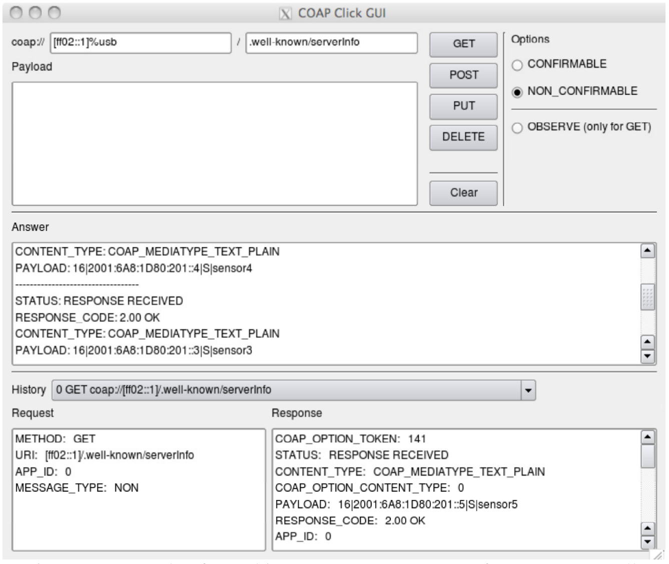


Figure 2. Discovery of sensor nodes by the sensor gateway by sending a CoAP multicast GET request for /.well-known/serverInfo in the sensor network



```
coap:// [ff02::1]%usb / .well-known/serverInfo
CONFIRMABLE [✓ NON_CONFIRMABLE]
OBSERVE (only for GET)

Answer
CONTENT_TYPE: COAP_MEDIATYPE_TEXT_PLAIN
PAYLOAD: 16|2001:6A8:1D80:201::4|S|sensor4
-----------------------------------
STATUS: RESPONSE RECEIVED
RESPONSE_CODE: 2.00 OK
CONTENT_TYPE: COAP_MEDIATYPE_TEXT_PLAIN
PAYLOAD: 16|2001:6A8:1D80:201::3|S|sensor3

History 0 GET coap://[ff02::1]/.well-known/serverInfo

Request
METHOD:  GET
URI:  [ff02::1]/.well-known/serverInfo
APP_ID:  0
MESSAGE_TYPE:  NON

Response
COAP_OPTION_TOKEN:  141
STATUS:  RESPONSE RECEIVED
CONTENT_TYPE:  COAP_MEDIATYPE_TEXT_PLAIN
COAP_OPTION_CONTENT_TYPE:  0
PAYLOAD:  16|2001:6A8:1D80:201::5|S|sensor5
RESPONSE_CODE:  2.00 OK
APP_ID:  0
```

Figure 3. Example of a multicast CoAP GET request for resource /.well-known/serverInfo as executed by the sensor gateway

send CoAP multicast requests for /.well-known/serverInfo in the LAN network in order to discover all sensor gateways. The resource /.well-known/serverInfo of a sensor gateway will also contain, in addition to the address and name of the sensor gateway, the domain suffix of the sensor subnet and the IPv6 prefix of the sensor subnet. In a similar way, the Internet gateway will add the address and name of the discovered sensor gateways to its local DNS. In addition, the Internet gateway will dynamically install a route to the sensor subnet and will add the sensor gateway as the name server for the sensor network. In case the Internet gateway notices that the sensor gateway does not have a subnet prefix, domain suffix and name configured, the Internet gateway will take this information from its pool (see our assumptions) and send it as a CoAP POST request to the sensor gateway, which will update its configuration accordingly.

This process can be repeated for different levels in the networking hierarchy up to the highest level, which, in our simple example, is the Internet gateway. Now, everyone in the Internet can resolve the FQDN of every discovered sensor and forward packets to this sensor. This is shown in Fig. 4, where the discovered sensor information consisting of the name “sensor3” and 16-bit address “3” has been used to create an IPv6 address and FQDN, which are dynamically added to the DNS on the sensor gateway.

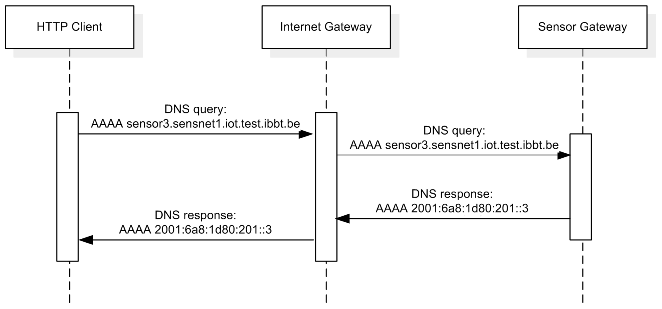

HTTP Client
Internet Gateway
Sensor Gateway
DNS query:
AAAA sensor3.sensnet1.iot.test.ibbt.be
DNS query:
AAAA sensor3.sensnet1.iot.test.ibbt.be
DNS response:
AAAA 2001:6a8:1d80:201::3
DNS response:
AAAA 2001:6a8:1d80:201::3

Figure 4. Name resolution using dynamically configured DNS information

This means that all sensors are now globally reachable with minimal effort and end-to-end communication is now possible. Using the same principles, one can introduce additional levels of indirection in order to enhance scalability or realize more complex setups. At this point, CoAP can be used to e.g. update the name of the sensor or retrieve any other information such as measurements.

3) *Discovery through hierarchy of linked CoAP servers*

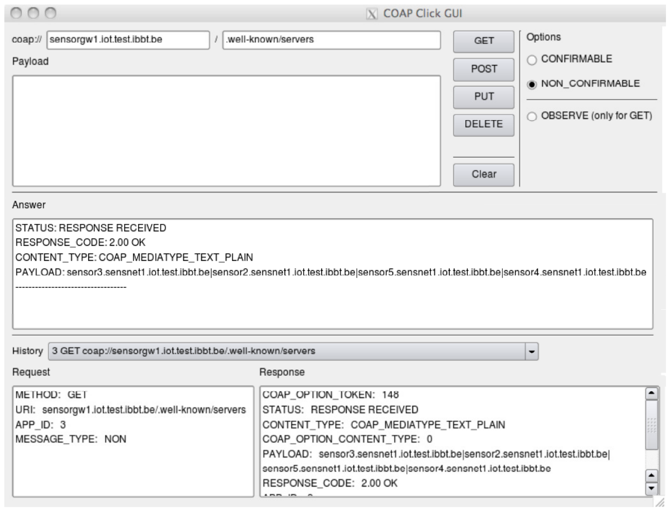
To further facilitate the discovery of sensors, we have introduced new CoAP resources for making available all discovered CoAP servers. As such, the sensor gateways and Internet gateway in our example will themselves run a CoAP server offering the following CoAP resources:

- /.well-known/servers: returns a list of all FQDNs for all discovered sensors
- /.well-known/servers/address: returns a list of all IPv6 addresses for all discovered sensors
- /.well-known/servers/coap: returns CoAP links to all discovered CoAP servers, using the CoRE Link Format (actually linking different CoAP servers together)

When a client wants to discover available sensors and make use of the services offered by sensor nodes, it now only has to know one anchor point for the entire domain of the organization (in a similar way a domain has a well-known name server). In our example, this is the Internet gateway, which could be assigned an easy to remember name such as coap.iot.test.ibbt.be. From that point on, a client can simply take the following actions:

- Send a CoAP request for the resource /.well-known/servers on the Internet gateway
- Per sensor gateway, the client can send a CoAP request to /.well-known/servers in order to find all sensors in the attached sensor subnet (see Fig. 5)
- Per sensor, the client can now use CoAP to retrieve sensor information

By applying this mechanism and creating a hierarchy of linked CoAP servers, any client can easily discover and use any sensor without a lot of network overhead. In



```
coap:// sensorgw1.iot.test.ibbt.be / .well-known/servers
CONFIRMABLE [✓ NON_CONFIRMABLE]
OBSERVE (only for GET)

Answer
STATUS: RESPONSE RECEIVED
RESPONSE_CODE: 2.00 OK
CONTENT_TYPE: COAP_MEDIATYPE_TEXT_PLAIN
PAYLOAD: sensor3.sensnet1.iot.test.ibbt.be|sensor2.sensnet1.iot.test.ibbt.be|sensor5.sensnet1.iot.test.ibbt.be|sensor4.sensnet1.iot.test.ibbt.be
-----------------------------------

History 3 GET coap://sensorgw1.iot.test.ibbt.be/.well-known/servers

Request
METHOD:  GET
URI:  sensorgw1.iot.test.ibbt.be/.well-known/servers
APP_ID:  3
MESSAGE_TYPE:  NON

Response
COAP_OPTION_TOKEN:  148
STATUS:  RESPONSE RECEIVED
CONTENT_TYPE:  COAP_MEDIATYPE_TEXT_PLAIN
COAP_OPTION_CONTENT_TYPE:  0
PAYLOAD:  sensor3.sensnet1.iot.test.ibbt.be|sensor2.sensnet1.iot.test.ibbt.be|
sensor5.sensnet1.iot.test.ibbt.be|sensor4.sensnet1.iot.test.ibbt.be
RESPONSE_CODE:  2.00 OK
```

Figure 5. Example of a client querying a sensor gateway for the FQDNs of all discovered sensors

combination with the automatic creation of FQDN names for sensors and their addition to a DNS system, this creates a flexible discovery mechanism and user-friendly access to sensors for humans. The whole process is almost fully automated, minimizing human intervention.

4) Summary

In Fig. 6, we summarize the entire self-organization process, discovery of sensors and access to resources. For simplicity, resolving of DNS names, as already illustrated in Fig. 4, is ignored. Also note that the CoAP servers, indicated between [], can be either DNS name or IPv6 address. Further, the order of discovery, first sensors and then sensor gateways, can be the other way around.

V. ENABLING HTTP ACCESS

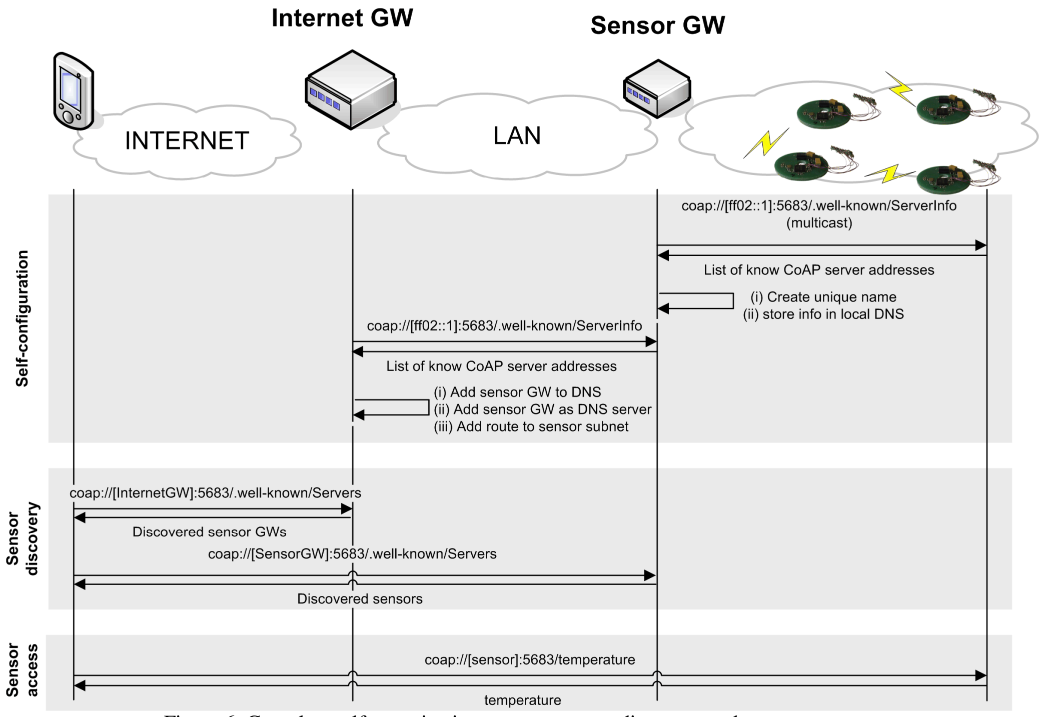
The proposed sensor resource self-configuration solution described in the previous section enables access to sensors using DNS names and IPv6 addresses using CoAP. However many clients do not have a CoAP implementation and will therefore not be able to benefit from this proposed solution. On the other hand all web client implementations have a web browser that supports HTTP. Since CoAP is following the same RESTful principles as HTTP, both protocols can be nicely mapped to each other and thus making the sensor resources accessible via HTTP. To achieve this mapping, HTTP-CoAP proxy functionality is required. In addition, to enable real browsable discovery of and access to sensor resources, we have foreseen a translation mechanism to create HTML pages from responses in the CoRE Link Format. Both mechanisms are explained in the following subsections.

A. HTTP-CoAP proxy functionality

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 and the client remains unaware of the fact that CoAP was used to retrieve the reply from the sensor.

The implemented proxy application on the gateways can act in two modes: transparent and non-transparent. In the non-transparent mode the client should construct the HTTP request in the following format: *http://gw_name:8080/sensor_name/resource*. Of course, the respective IPv6 addresses can be used instead of the names in the above format. The gateway then translates this request into the following CoAP request and sends it to the respective sensor: *coap://sensor_name/resource*. It is clear that in this non-transparent mode, the client must explicitly be aware of the proxy and use it as part of the URI.

In the transparent mode the client remains unaware of the presence of the proxy functionality on the gateway and constructs the HTTP request in the following format: *http://sensor_name:8080/resource*. For this proxy to work properly in transparent mode, the proxy has to be on the path between the client and the sensor in order to be able to intercept the HTTP request (and TCP connection) and map it into the appropriate CoAP request. In our example, the transparent proxy functionality for accessing the sensors


Internet GW
Sensor GW
INTERNET
LAN
Self-configuration
coap://[ff02::1]:5683/.well-known/ServerInfo
(multicast)
List of know CoAP server addresses
(i) Create unique name
(ii) store info in local DNS
coap://[ff02::1]:5683/.well-known/ServerInfo
List of know CoAP server addresses
(i) Add sensor GW to DNS
(ii) Add sensor GW as DNS server
(iii) Add route to sensor subnet
Sensor discovery
coap://[InternetGW]:5683/.well-known/Servers
Discovered sensor GWs
coap://[SensorGW]:5683/.well-known/Servers
Discovered sensors
Sensor access
coap://[sensor]:5683/temperature
temperature
Figure 6. Complete self-organization process, sensor discovery and resource access

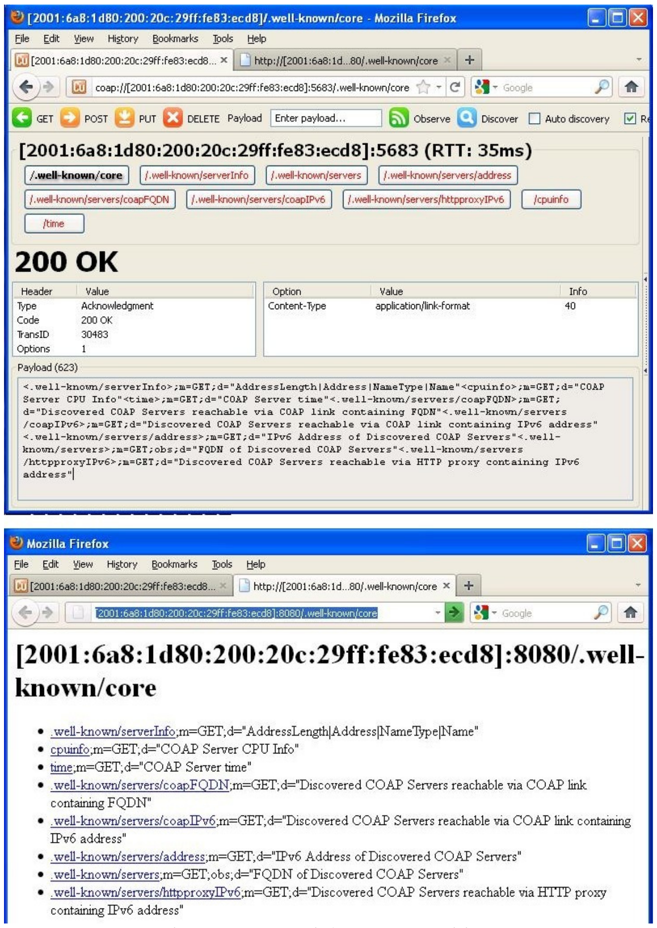
resides only on the sensor gateways. When the HTTP request for the sensor enters, the proxy will behave as the end point of the TCP connection and will handle the TCP connection. In the background, a translation to CoAP takes place and the request is sent to the sensor. For the user it seems as if he connects directly to the sensor using HTTP/TCP, but in reality the sensor gateway transparently handles the connection and translates it to CoAP. As such, in transparent mode, the user does not have to be aware of a proxy that it needs to use.

B. *CoRE Link Format to HTML*

In addition to the mapping between HTTP and CoAP, the proxy implementation on the gateway 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. Every resource in a response in the CoRE Link Format, such as </sensors/temp> is rewritten by the proxy into an HTML link. When the original request made use of a proxy, the HTTP URI will consist of the proxy address or name, followed by the address or name of the actual CoAP server on which the resource resides and the resource itself. When the original request did not make use of a proxy or transparent proxying is possible, the HTTP URI will only contain the actual CoAP server on which the resource is located followed by the resource. In Fig. 7 an example is shown of how the CoRE Link Format is automatically rewritten into an HTML page. On top, the result of the CoAP request in CoRE Link Format is shown. Below, the HTML page that has been automatically created by the HTTP-CoAP proxy is shown.

VI. Implementation and Deployment

Together, the mechanism described in Sections IV and V, realize a hierarchical self-configuration solution based on CoAP. Automatically discovered CoAP servers, up to the sensor level, are linked together into a browsable hierarchy that can be accessed either via CoAP or HTTP, offering global access to sensor resources in a human-friendly way through the use of names. The described solutions have been implemented and deployed on a publicly reachable testbed (IPv6 only), of which the details are shown in Fig. 8. Anyone that is connected to the Internet using IPv6, is able to access


[2001:6a8:1d80:200:20c:29ff:fe83:ecd8]/.well-known/core - Mozilla Firefox
File Edit View History Bookmarks Tools Help
[2001:6a8:1d80:200:20c:29ff:fe83:ecd8...
http://[2001:6a8:1d...80/.well-known/core
coap://[2001:6a8:1d80:200:20c:29ff:fe83:ecd8]:5683/.well-known/core
Google
GET POST PUT DELETE Payload Enter payload... Observe Discover Auto discovery

[2001:6a8:1d80:200:20c:29ff:fe83:ecd8]:5683 (RTT: 35ms)

/.well-known/core | /.well-known/serverInfo | /.well-known/servers | /.well-known/servers/address | /.well-known/servers/coapFQDN | /.well-known/servers/coapIPv6 | /.well-known/servers/httpproxyIPv6 | /cpuinfo | /time

200 OK

Header	Value
Type	Acknowledgment
Code	200 OK
TransID	30483
Options	1

Option	Value	Info
Content-Type	application/link-format	40

Payload (623)

```
<.well-known/serverInfo>;m=GET;d="AddressLength|Address|NameType|Name"<cpuinfo>;m=GET;d="COAP
Server CPU Info"<time>;m=GET;d="COAP Server time"<.well-known/servers/coapFQDN>;m=GET;
d="Discovered COAP Servers reachable via COAP link containing FQDN"<.well-known/servers
/coapIPv6>;m=GET;d="Discovered COAP Servers reachable via COAP link containing IPv6 address"
<.well-known/servers/address>;m=GET;d="IPv6 Address of Discovered COAP Servers"<.well-
known/servers>;m=GET;obs;d="FQDN of Discovered COAP Servers"<.well-known/servers
/httpproxyIPv6>;m=GET;d="Discovered COAP Servers reachable via HTTP proxy containing IPv6
address"
```

Mozilla Firefox
File Edit View History Bookmarks Tools Help
[2001:6a8:1d80:200:20c:29ff:fe83:ecd8...
http://[2001:6a8:1d...80/.well-known/core
[2001:6a8:1d80:200:20c:29ff:fe83:ecd8]:8080/.well-known/core
Google

[2001:6a8:1d80:200:20c:29ff:fe83:ecd8]:8080/.well-known/core

- .well-known/serverInfo;m=GET;d="AddressLength|Address|NameType|Name"
- cpuinfo;m=GET;d="COAP Server CPU Info"
- time;m=GET;d="COAP Server time"
- .well-known/servers/coapFQDN;m=GET;d="Discovered COAP Servers reachable via COAP link containing FQDN"
- .well-known/servers/coapIPv6;m=GET;d="Discovered COAP Servers reachable via COAP link containing IPv6 address"
- .well-known/servers/address;m=GET;d="IPv6 Address of Discovered COAP Servers"
- .well-known/servers;m=GET;obs;d="FQDN of Discovered COAP Servers"
- .well-known/servers/httpproxyIPv6;m=GET;d="Discovered COAP Servers reachable via HTTP proxy containing IPv6 address"

Figure 7. CoRE Link Format rewriting.
On top, the result of a CoAP request in CoRE Link Format.
Below, the automatically created HTML page.

the resources, either over CoAP or HTTP (port 8080).

The implementation consists of two parts, the implementation running on the sensors nodes and the implementation on the gateways. The implementation on the gateways has been realized in Click Router, a C++ based modular framework that can be used to realize any network packet processing functionality [16]. It consists of several modules such as the CoAP protocol, a CoAP server backend

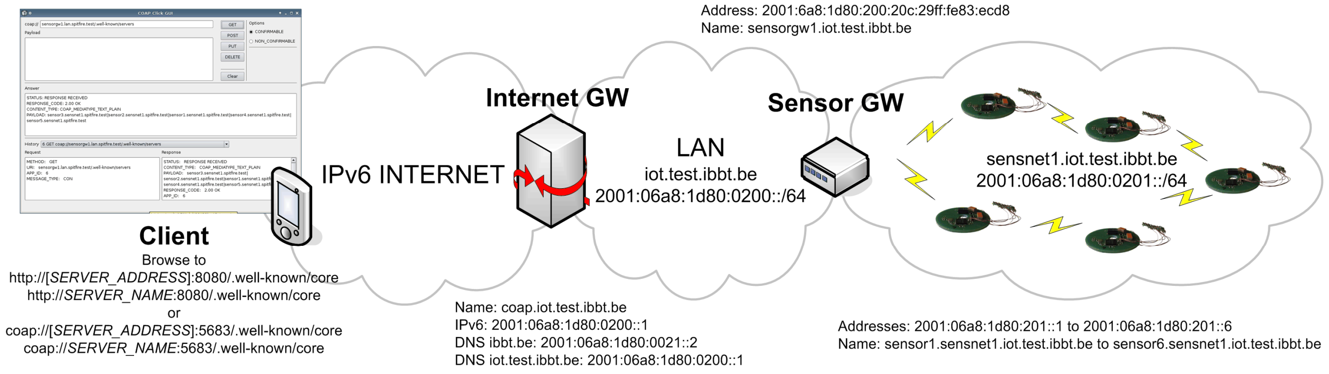

Address: 2001:6a8:1d80:200:20c:29ff:fe83:ecd8
Name: sensorgw1.iot.test.ibbt.be
Internet GW
Sensor GW
LAN
iot.test.ibbt.be
2001:06a8:1d80:0200::/64
IPv6 INTERNET
sensnet1.iot.test.ibbt.be
2001:06a8:1d80:0201::/64
Client
Browse to
http://[SERVER_ADDRESS]:8080/.well-known/core
http://SERVER_NAME:8080/.well-known/core
or
coap://[SERVER_ADDRESS]:5683/.well-known/core
coap://SERVER_NAME:5683/.well-known/core
Name: coap.iot.test.ibbt.be
IPv6: 2001:06a8:1d80:0200::1
DNS ibbt.be: 2001:06a8:1d80:0021::2
DNS iot.test.ibbt.be: 2001:06a8:1d80:0200::1
Addresses: 2001:06a8:1d80:201::1 to 2001:06a8:1d80:201::6
Name: sensor1.sensnet1.iot.test.ibbt.be to sensor6.sensnet1.iot.test.ibbt.be

Figure 8. Publicly reachable testbed at IBBT

capable of offering resources, CoAP server discovery, HTTP-CoAP proxying, USB sensor communication… These modules can be combined in several ways by creating a configuration file. As such, using the same code base, one can realize the following configurations: a) a stand-alone socket-based CoAP client making use of IPv6/UDP sockets for network communication b) a stand-alone packet-based CoAP client that processes and generates complete network packets (including Ethernet/IPv6/UDP headers) offering full control over the communication which is interesting for the realization of the gateway functionality c) a CoAP sensor gateway with sensor discovery, DNS and (non-)transparent proxy functionality d) a CoAP internet gateway with sensor gateway discovery, DNS and non-transparent proxy functionality.

The CoAP client/server protocol and the CoAP resources on the sensors have been implemented using the IDRA framework [17]. IDRA is a network architecture and application platform developed for TinyOS and written in nesC. The designed solution (including MAC, AODV routing and CoAP) has a footprint of 37092 bytes in ROM and 5923 bytes in RAM.

VII. Related work

In this section we will discuss other work focusing on the automatic discovery of sensors, realization of end-to-end access and integration with DNS. Our solution is a network-based solution, meaning that we want to achieve global end-to-end access to sensor resources in a way that requires minimal to no human intervention. At the same time, we want to comply with current Web Standards by offering access using DNS and HTTP and we want to foresee means for the automatic discovery of sensors within a domain, since global access alone is not enough. Users should be able to find out which sensors are available.

Some solutions focus on the discovery of sensors and sensor data by publishing or collecting information about the sensors and measurements in databases that can be accessed by other applications and services. For example, Pachube [18] allows sensor data to be pushed to a central database, where it can be used by others to create applications. No direct access to sensors is allowed. The OGC SWE framework [19] defines web service interfaces for accessing sensor data, controlling sensors and alerting, functionalities comparable to the ones offered by CoAP. Reference [20] presents the OSIRIS Sensor Web Discovery Framework, which makes use of registries that are being build and which are capable of handling the dynamic properties of sensors. Similarly, a web crawler could periodically scan the Web of Things for sensors and downloading meta-data via their RESTful interfaces. This information can be stored, e.g. as RDF triples, after which the information can be searched, reasoned upon or linked with other open data [21].

These solutions aim for the realization of a Semantic Web of Things or Sensor Web, which enables data producers and users to publish and access sensor information via web- and standards based interface (see [22] for more details on Sensor Web). This goes already one step beyond our solution, since it focuses on the service part, skipping the deployment steps and ignoring the networking part. For example, aspects such as naming, automatic routing to sensor subnets, facilitating the deployment of sensor networks… are not considered, although very important. Our solution provides an answer to these problems, can be seen as a building block for the realization of the Semantic Web of Things and is therefore quite different, but complementary.

When focusing now more on the networking aspect (discovery followed by integration in DNS and automatic routing to sensor subnets), few related work is found. In [23], a solution is presented where a lightweight mDNS-SD implementation is running on a sensor platform. As such, sensor nodes can announce their services and update resource records in a DNS. However, this solution does not include other aspects such as the discovery and self-organization at higher hierarchical levels or the automatic configuration of routing to the subnet. Further, if RESTful web services want to be offered on top of the discovery, both an mDNS-SD and CoAP implementation are needed, increasing the code footprint. Taking a RESTful approach to tackle both problems mitigates this problem. In [24], a solution is presented for the integration of sensors and actuators in the Future Internet in a plug and play manner. Sensor nodes can register with gateways that provide an open interface to access raw or abstracted sensor data. At a higher level, a Sensor Address Server maintains a list of all registered gateways. This solution provides hierarchical levels, but does not comply with existing Internet standards nor foresees direct sensor resource access using IPv6. Finally, in [25], a zero-configuration IPv6/6LoWPAN-based system architecture is described. It foresees an API to access services following REST principles. A central unit can make use of this API to auto-discover the functionality offered by the sensor node or the service can be advertised in a way similar to mDNS. This approach uses embedded web services (not CoAP), but does not achieve the level of auto-configuration from our solution, i.e. multiple self-organizing hierarchical levels automatically linked with each other, resulting in a browsable discovery system that allows the discovery of and access to sensor resources. Also DNS aspects or automatic routing to sensor subnets, crucial in the actual roll-out of the network, are not considered.

VIII. Conclusions and outlook

In this paper we have described a novel self-organization solution 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 embedded web service technology, i.e. the IETF CoAP protocol. By combining it with DNS and foreseeing HTTP-CoAP proxy functionality, it complies with current Internet standards. Automatic hierarchical discovery of CoAP servers is one of the key features, resulting in a browsable hierarchy of CoAP servers, up to the level of the sensor resources. By creating a hierarchy of linked CoAP servers, scalability can be addressed. As such, the proposed approach provides a feasible and flexible solution to achieve hierarchical self-organization with a minimum of pre-configuration. The solution is based on a minimal number of assumptions regarding the pre-configuration. With some additional improvements and the development of management tools, it provides a valuable contribution to facilitate the deployment of and access to sensor networks.

The fact that embedded web services are used is a strong point, since it will facilitate integration with other services and applications. By complementing the solution with the appropriate firewalling and access policies, any level of sensor access can be made possible. The implementation and proper functioning of the solution has been demonstrated through the deployment on a publicly accessible test setup. In the future, a detailed analysis of the overhead of the presented solution will take place, together with a comparison between the pull-based approach presented here and a push-based approach where sensors announce their presence to the sensor gateway. This push-based approach has been implemented already and the evaluation of both solutions is currently ongoing using real-life wireless sensor network testbed, namely w-iLab.t [26]. This evaluation will give us further insights about the strengths and limitations of the proposed approach in a large-scale, real-life environment. Next to this the scalability and external accessibility will be further explored and tested and issues such as the security of the presented solution will be investigated e.g. through the use of DTLS. In addition, it will be investigated to what extend any manual configuration that is still required can be avoided and how management tools based on embedded web service technology can bring the self-organization, configuration and management of sensor networks to a next level.

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) and from the IBBT ICON project GreenWeCan.

References

[1] N. Kushalnagar, G. Montenegro, and C. Schumacher, IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): 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] Z. Shelby, B. Frank, and D. Sturek, Constrained Application Protocol (CoAP), draft-ietf-core-coap-07, work in progress, July 2011.

[6] Carnot Institute, White paper: Smart Networked Objects and Internet of Things, http://www.instituts-carnot.eu/files/AiCarnot-White_Paper-Smart_Networked_Objects_and_Internet_of_Things.pdf

[7] J. Zheng and A. Jamalipour, Wireless Sensor Networks – A Networking Perspective, Wiley, 2009.

[8] J. Hui and D. Culler, "IP is dead, long live IP for wireless sensor networks," Proc. 6th ACM Conference on Embedded Network Sensor Systems, pp. 15-28, 2008, doi:10.1145/1460412.1460415.

[9] M. Chen, S. Mao, Y. Xiao, M. Li, and V. Leung, "IPSA: A Novel Architecture Design for Integrating IP and Sensor Networks," International Journal of Sensor Networks (IJSNet), Vol. 5, No. 1, 2009, pp. 48- 57, DOI: 10.1504/IJSNET.2009.023315.

[10] S. Duquennoy, N. Wirstom, N. Tsiftes, and A. Dunkels, "Leveraging IP for Sensor Network Deployment," Proc. workshop on Extending the Internet to Low power and Lossy Networks, 2011.

[11] J-P. Vasseur and A. Dunkels, Interconnecting Smart Objects with IP – The Next Internet, ISBN 978-0123751652, Morgan Kaufmann, 2010.

[12] 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, doi:10.1145/1810279.1810289.

[13] 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.

[14] Z. Shelby, CoRE Link Format, draft-ietf-core-link-format-07, work in progress, July 2011.

[15] Z. Shelby and C. Bormann, 6LoWPAN – The Wireless Embedded Internet, Chapter 3, ISBN 978-0-470-74799-5, Wiley, 2009.

[16] E. Kohler, R. Morris, B. Chen, J. Jannotti and M. F. Kaashoek. "The Click modular router," ACM Transactions on Computer Systems 18(3), pp. 263-297, 2000, doi:10.1145/354871.354874.

[17] E. De Poorter, E. Troubleyn, I. Moerman and P. Demeester, "IDRA: a Flexible System Architecture for Next-Generation Wireless Sensor Networks," Wireless Networks, 17(6), pp. 1423-1440, 2011, doi: 10.1007/s11276-011-0356-5

[18] Pachube, http://www.pachube.com.

[19] M. Botts, G. Percivall, C. Reed, and J. Davidson, OGC White Paper - OGC Sensor Web Enablement: Overview And High Level Architecture, Open Geospatial Consortium Inc, 2007.

[20] S. Jirka, A. Broring, and C. Stasch, "Discovery Mechanisms for Sensor Web," Sensors, 9(4), pp. 2661-2681, 2009, doi:10.3390/s90402661.

[21] D. Pfisterer et al., "SPITFIRE: Towards a Semantic Web of Things," IEEE Communications Magazine Special Issue on the Internet of Things, pp. 40-48, Nov. 2011.

[22] T. Foerster, D. Nüst, A. Bröring, and S. Jirka, "Discovering the Sensor Web through Mobile Applications," Proc. 8th International Symposium on Location-Based Services, 2011.

[23] 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.

[24] J. Schneider, A. Klein, C. Mannweiler, and H. D. Schotten, "An efficient architecture for the integration of sensor and actuator networks into the future internet," Advances in Radio Science, 9, pp. 231-235, 2011, doi:10.5194/ars-9-231-2011.

[25] L. Schor, P. Sommer, and R. Wattenhofer, "Towards a Zero-Configuration Wireless Sensor Network Architecture for Smart Buildings," Proc. First ACM Workshop On Embedded Sensing Systems For Energy-Efficiency In Buildings, 2009, doi:10.1145/1810279.1810287.

[26] S. Bouckaert, W. Vandenberghe, B. Jooris, I. Moerman, and P. Demeester, "The w-iLab.t testbed," Proc. of the International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCom '10), pp. 145–154, Berlin, Germany, May 2010.