Enabling the web of things: facilitating deployment, discovery and resource access to IoT objects using embedded web services

Isam Ishaq*, Jeroen Hoebeke, Jen Rossey, Eli De Poorter, Ingrid Moerman and Piet Demeester

Department of Information Technology (INTEC),
Ghent University – iMinds,
Ghent, Belgium
Email: isam.ishaq@intec.ugent.be
Email: jeroen.hoebeke@intec.ugent.be
Email: jen.rossey@intec.ugent.be
Email: eli.depoorter@intec.ugent.be
Email: ingrid.moerman@intec.ugent.be
Email: piet.demeester@intec.ugent.be
*Corresponding author

Abstract: Today, the IETF Constrained Application Protocol (CoAP) is being standardised. CoAP takes the internet of things to the next level: it enables the implementation of RESTful web services on embedded devices, thus enabling the construction of an easily accessible web of things. However, before tiny objects can make themselves available through embedded web services, 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-organisation solution to facilitate the deployment of constrained networks and enable the discovery, end-to-end connectivity and service usage of these newly deployed sensor nodes. By using embedded web service technology, the need of other protocols on these resource constrained devices is avoided. It allows automatic hierarchical discovery of CoAP servers, resulting in a browsable hierarchy of CoAP servers, which can be accessed both over CoAP and hypertext transfer protocol.

Keywords: CoAP; self-organisation; IoT; internet of things; WoT; web of things; DNS; domain name system; proxy; embedded web services; discovery.

Reference to this paper should be made as follows: Ishaq, I., Hoebeke, J., Rossey, J., De Poorter, E., Moerman, I. and Demeester, P. (2014) 'Enabling the web of things: facilitating deployment, discovery and resource access to IoT objects using embedded web services', *Int. J. Web and Grid Services*, Vol. 10, Nos. 2/3, pp.218–243.

Biographical notes: Isam Ishaq is currently the Director of the Said Khoury IT Center of Excellence (SKITCE) at Al-Quds University and a PhD candidate at the University of Ghent. He obtained his graduation degree in Electrical Engineering from the Technical University of Berlin in 1996. He was technical director of the Palestinian Academic Network (PLANET) and one of the

biggest Palestinian ISPs (Palestine Online). His research interests include mobile and wireless networks and the realisation of the internet of things. He has had publications in international journals and conference proceedings and has contributed to internet drafts.

Jeroen Hoebeke received the Master's degree in Engineering (Computer Science) from the University of Ghent in 2002. Since August 2002, he has been affiliated with the Internet-based Communication Networks and Services Research Group (IBCN) that belongs to both Ghent University and the Flemish research institute iMinds. He obtained a PhD in 2007 on Adaptive Ad Hoc Routing and Virtual Private Ad Hoc Networks. He is currently postdoctoral researcher where he is conducting research on mobile and wireless networks. His main focus is on network architectures and protocols for realising the Internet of Things.

Jen Rossey received the Master's degree in Applied Engineering Sciences (Computer Science) from University College Ghent (HoGent) in 2010. Since March 2010, he has been affiliated with the Broadband Communications Networks research group (IBCN) at the Department of Information Technology in Ghent University where he is currently working as a Researcher. He is conducting research on mobile and wireless networks, future internet and sensor networks. He has been involved in several national and international research projects targeting a range of application domains, such as indoor positioning, machine-to-machine communication, e-health.

Eli De Poorter received his Master's and PhD degrees in Computer Engineering from the Ghent University, Belgium, in 2006 and 2011, respectively. He is the Creator of the IDRA architecture, a flexible framework for heterogeneous networked devices. He is currently working as a FWO postdoctoral researcher and project coordinator at the IBCN group of the same university. His main research interests include all networking aspects of the internet of things (IoT), wireless communications, network architectures, wireless sensor and ad hoc networks, the future internet and next-generation network architectures.

Ingrid Moerman received her degree in Electrical Engineering (1987) and the PhD degree (1992) from the Ghent University, where she became a part-time professor in 2000. She is a staff member of the Internet-based Communication Networks (IBCN) and services research group, where she is leading the research on mobile and wireless communication networks. Since 2006 she joined the Flemish research institute iMinds, where she is coordinating several interdisciplinary research projects. She is author or co-author of more than 600 publications in international journals or conference proceedings.

Piet Demeester is Professor in the faculty of Engineering at Ghent University. He is Head of the research group 'Internet-Based Communication Networks and services' (IBCN) that is part of the Department of Information Technology of Ghent University. The group is focusing on several advanced research topics: network modelling, design and evaluation; mobile and wireless networking; high-performance multimedia processing; autonomic computing and networking; service engineering; content and search management and data analysis and machine learning. He is also leading the future internet (networks, media and service) Department of the Flemish research institute iMinds. He is Fellow of the IEEE.

This paper is a revised and expanded version of a paper entitled 'Facilitating sensor deployment, discovery and resource access using embedded web services' presented at the 'Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)', Palermo, Italy, 4–6 July 2012.

1 Introduction

The ubiquitous internet protocol technology is rapidly spreading in 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 1990s 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. Most of these efforts were focusing on the networking layer: IPv6 over low-power wireless personal area networks (RFC4919) (Kushalnagar et al., 2007), transmission of IPv6 packets over IEEE 802.15.4 networks (RFC4944) (Montenegro et al., 2007), RPL: IPv6 routing protocol for low-power and lossy networks routing (RFC6550) (Winter et al., 2012) or the ZigBee adoption of IPv6 ('ZigBee IP', n.d.). These standardisation efforts enable the realisation of an internet of things, where end-to-end connectivity with tiny objects such as sensors and actuators becomes possible.

However, the great success of the current internet was not caused by solely supporting global connectivity. Only with the advent of web service technology, the development of all kinds of interesting services became possible and the world wide web became a reality. Today (RESTful), web service technology is at the basis of many successful companies. A similar embedded counterpart of web service technology is needed in order to exploit all great opportunities offered by the IoT and turn it into a Web of Things (WoT). Recently, standardisation work has started to allow precisely the integration of constrained devices with the internet at the service level. The IETF Constrained RESTful Environments (CoRE) working group is in the process of realising 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 specialised RESTful web transfer protocol for use with constrained networks and nodes (Shelby et al., 2012). CoAP realises a subset of REST that is common with the Hypertext Transfer Protocol (HTTP), but is optimised for Machine-to-Machine (M2M) applications.

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. Within the sensor network itself, the available protocols are largely self-organising, 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 Domain Name System (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, for example, sensors and sensor gateways, offers new opportunities to tackle several of these challenges related to the deployment of sensor networks and the realisation 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 between HTTP and 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-organisation 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.

Section 2 discusses how the ongoing work in the IETF CoRE working group contributes to the realisation of the WoT. Next, we give an overview of the challenges related to sensor network deployment, discovery and access (Section 3). In Section 4, we then introduce a solution based on CoAP and DNS for the hierarchical self-configuration of sensors and access via HTTP, followed by a brief discussion in Section 5 about all new possibilities that are unleashed when tiny objects can be easily deployed, integrated with the internet and made accessible via embedded web service technology. Section 6 presents actual deployment results and in Section 7 the performance of the deployment is analysed. Section 8 gives an overview of related work. The paper is concluded in Section 9.

2 From IoT to WoT

Recent research on embedded web services is laying the groundwork for a better integration of sensor resources into the service web and, as such, provides the foundations for realising what is called the WoT. Since the dominating web protocol HTTP is too complex, the IETF CoRE working group, formed in 2010, has designed a simpler web protocol – CoAP. CoAP uses the same RESTful principles as HTTP, but it is much lighter so that it can be run on constrained devices (Colitti et al., 2011b; Yazar and Dunkels, 2009). 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.

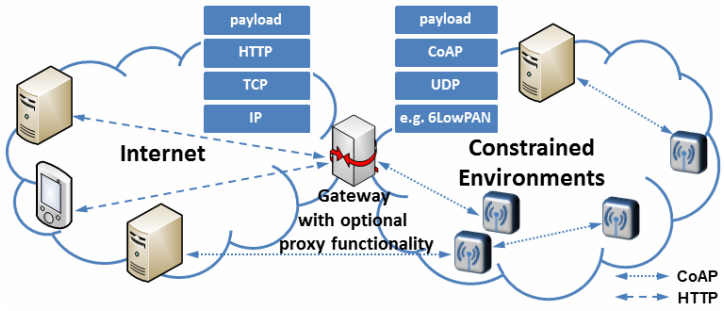
At the time of writing this paper, the CoAP protocol was not finalised yet. However it is considered in its final stages before being finalised. Nevertheless several implementations of the CoAP protocol for various platforms and programming languages already exist; many of which run on Classes 1 and 2 devices as classified by Bormann (2012). Class 1 devices are resource-constrained devices with ~10 KB of RAM, and ~100 KB of ROM. Class 2 devices have ~50 KB of RAM and ~250 KB of ROM. Interoperability between many of these implementations has been formally tested by the *European Telecommunications Standards Institute* (ETSI) at two events called ETSI IoT CoAP Plugtests. In addition to assessing the interoperability of participating products, these Plugtests events aimed to validate the CoRE base standards. The existence of such a wide range of implementations across a broad range of programming languages and most importantly platforms demonstrates the feasibility of the protocol implementation. For an overview of the CoAP-based protocol and related drafts we refer to the study of Ishaq et al. (2013).

CoAP and HTTP have been compared several times in the literature. For example, Colitti et al. (2011a) compared CoAP to HTTP in terms of mote's energy consumption by simulations for a fixed 10 s client request interval. The results show that while receiving and processing packets, the energy consumed when using CoAP is approximately half compared to the one consumed when using HTTP. While transmitting packets, the energy required by CoAP was four times lower than the energy required by HTTP. When testing the response time on real-sensor motes, this study shows that using CoAP over UDP introduces 9- to 10-fold lower response times than HTTP over TCP. Similarly Pötsch (2011) reports that CoAP/UDP perform better than HTTP/TCP for the intelligent cargo container use case that was evaluated. In particular, the author reports a six times lower message size and a four times lower Round Trip Time (RTT). This is mainly due to CoAPs compressed header and the avoidance of the TCP handshaking mechanisms. The study of Kuladinithi et al. (2011) shows that generally UDP-based protocols perform better for constrained networks due to using lower number of messages when retrieving resources. But even when both protocols (CoAP and HTTP) are run over UDP, the same study shows that CoAP performs better than HTTP in constrained environments. CoAP also has the added value of providing optional reliability since it has its own simple retransmission capability.

The IETF CoRE working group considers CoRE as an extension of the current web architecture as illustrated in Figure 1. 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 (Shelby, 2010). 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. This proxy functionality is often implemented on the gateways that interconnect the constrained environments to the internet. Thus, gateways play a central role in the constrained environments architecture as can be seen in Figure 1. These

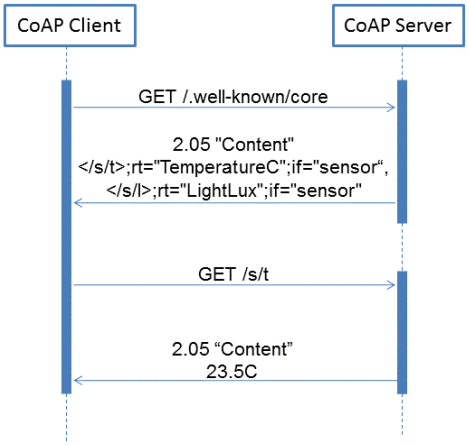
gateways have to be able to communicate between the internet protocol stack and the constrained environments protocol stack (see Figure 1) and to translate between them as needed. In the remainder of this paper, we will use the term *gateway* (GW) for the device that is located at border of constrained network and providing routing and communication functionality. We will use the term *proxy* whenever we talk about translation between protocols or whenever data are being cached.

Figure 1 Constrained environments architecture and the CoAP protocol stack (see online version for colours)



Resource discovery is important for M2M interactions and is supported in CoAP using the CoRE Link Format (Shelby, 2012). 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 CoAP 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 a 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 (23.5°C).

Figure 2 An example of resource discovery and access using CoAP (see online version for colours)



3 Problem statement

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 (IoT-A, 2011) and potentially, calibration. 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 communicating to and from the sensor gateway. Looking at the literature and standardisation bodies (see Section 1), it is clear that there have been many efforts to create such protocols, including MAC layer protocols, addressing, routing, data collection protocols, etc. (Zheng and Jamalipour, 2009). 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 with proxy functionality to be able to translate from and to the sensor network protocols. On the other hand, there is significant momentum for IP-based sensors and actuators as illustrated by the IETF work mentioned in Section 1 and in 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 (e.g. Chen et al., 2009; Duquennoy et al., 2011; Hui and Culler, 2008; Vasseur and Dunkels, 2010).

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. 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.

4 Solution

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.

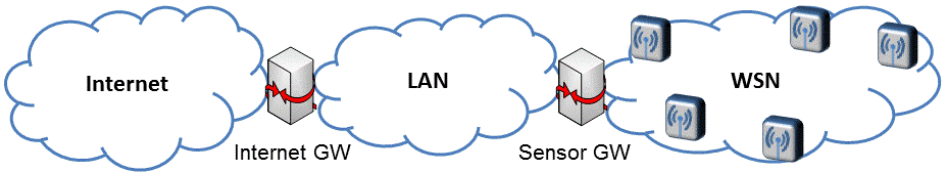
4.1 Assumptions

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.

To present our self-configuration solution, we assume a basic network topology as shown in Figure 3 (more complex topologies are possible, but are out of the scope of this paper). An organisation 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 organisation wants to equip its building(s) with wireless sensors, which will be connected to this 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.

Figure 3 Sample network topology used to present the self-organisation solution. A Wireless Sensor Network (WSN) is connected via a sensor gateway to a Local Area Network (LAN), which in turn is connected to the internet via an internet gateway (see online version for colours)

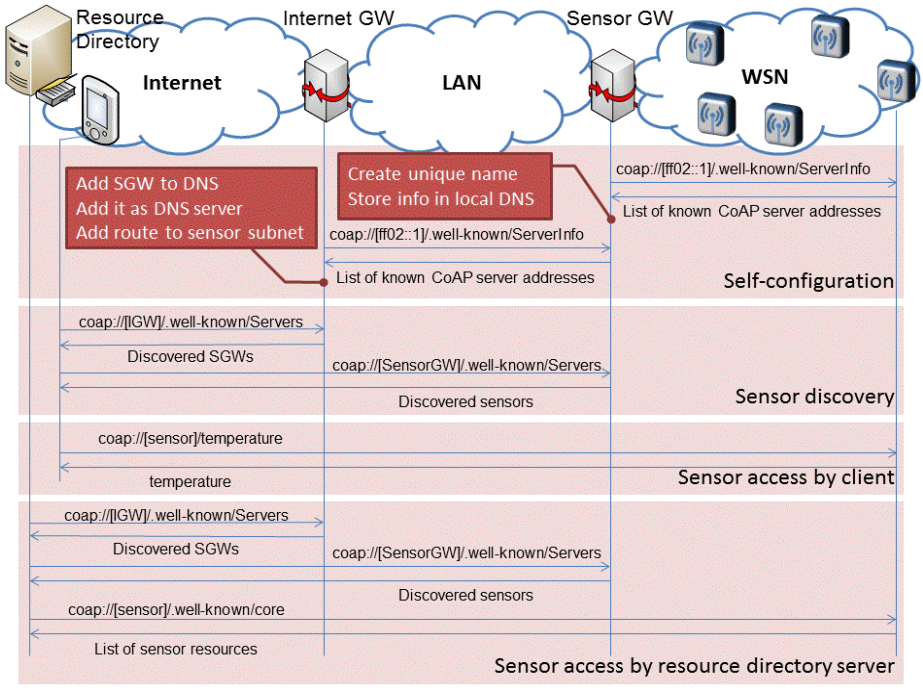


We consider the sensors as dumb devices that only have a minimal knowledge. For our self-configuration solution, we make the following assumptions. The sensor knows or will discover its address in the sensor network. Typically, because of the limitations of a sensor device, these addresses are preferably small, e.g. only 16-bit for 6LoWPAN short addresses (Shelby and Bormann, 2009). 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 unique, manually configured 16-bit addresses for the sensors. The assignment or generation of these unique addresses is out of the scope of this paper. Further, we assume that the sensor knows its (or a) name. This name is a variable-length string and could be anything, 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 standardised 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.

4.2 The solution

Based on these assumptions, we have designed a complete self-organisation solution which is summarised in Figure 4. Please note that for simplicity, resolving of DNS names is ignored in this figure. Also note that the CoAP servers, indicated between [], can use either DNS names or IPv6 addresses. As can be seen in this figure, our solution is organised in three phases: self-configuration, sensor discovery and sensor access (by a client or by a resource directory server).

Figure 4 Complete self-organisation process, sensor discovery and resource access by a client and by a resource directory server (see online version for colours)



After the sensors have been deployed, the sensor gateway discovers the sensors in its network by sending a multicast (ff02::1) CoAP GET request for the resource /.well-known/serverInfo. The response sent by this resource is currently a plain text string that contains whatever information the sensor knows about itself according to the following proprietary format:

`AL|`*`addressLength`*`|A|`*`address`*`|NT|`*`nameType`*`|N|`*`name`*`|SID|`*`subnetID`*
`|SP|`*`prefix`*`|SM|`*`subnet`*`|SD|`*`domain`*

Each sensor replaces the italic texts in the above format with its own information. As a minimum the sensor would send its address length (AL), its address (A), its name type (NT, S for short names and L for long names), and its name. The other information describes the subnet in which the sensor is located and might or might not be known by the sensor itself. Ideally, in the future, a standardised format should be used, possibly using a structured representation such as JSON.

When receiving the replies from the sensors to the multicast request, the sensor gateway is able to discover the (short) address and name of all sensor nodes present in the network. To verify whether already discovered sensors have disappeared (or new ones have been deployed), the sensor gateway can be configured to periodically perform the multicast. Of course, in this case the multicast frequency should be limited in order to limit the resulting energy consumption. Since discovery is triggered by the gateway, we refer to this approach as a pull-based solution. Section 7 will describe an alternative method we have implemented where sensor nodes announce their presence to the gateway, once or periodically (push versus pull-based solution).

Assuming 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 each 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 requests for names in the sensor domain). When a sensor is no longer available, i.e. it does not reply to a certain number (3 by default) of consecutive periodic discovery broadcasts, its 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.

The same discovery process can be repeated at a higher level in the network hierarchy assuming that the sensor gateways also run CoAP servers. The internet gateway can periodically 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 adds the address and name of the discovered sensor gateways to its local DNS. In addition, the internet gateway dynamically installs a route to the sensor subnet and adds 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. Please note that the order of discovery, first sensors and then sensor gateways, can be the other way around.

This discovery 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 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 realise more complex set-ups. At this point, CoAP can be used to, for example, update the name of the sensor or retrieve any other information such as measurements.

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 organisation (in a similar way a domain has a well-known name server). In our simple example, this is the internet gateway, which could be assigned an easy way to remember name such as coap.iot.test.ibbt.be. From that point, a client can simply take the following actions:

- Send a CoAP request for the resource /.well-known/servers on the internet gateway to get a list of all sensor gateways.
- 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.
- 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 (a human, another machine or a resource directory server) can easily discover and use any sensor without a lot of network overhead. For example, Figure 4 also shows how a resource directory can benefit from our solution in order to create a directory listing containing all embedded resources that are present in the network. In combination with the automatic creation of FQDN names for sensors and their addition to a DNS system, this creates a flexible discovery mechanism and enables user-friendly access to sensors for humans. The whole process is almost fully automated, minimising human intervention.

4.3 HTTP access to sensors

The solution as described thus far 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 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 (Castellani et al., 2012) and thus making the sensor resources accessible via HTTP. To achieve this mapping, HTTP-to-CoAP proxy functionality is required. In addition, to enable real browsable discovery 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 this subsection.

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-to-CoAP proxies capable of translating HTTP messages to CoAP messages and vice versa. Clients can access these gateways via their favourite 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. 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 future, we also want to foresee non-transparent proxying based on the CoAP Proxy-Uri option as described in the work of Shelby et al. (2012). Using this method, the CoAP (or HTTP) request is sent to the proxy and the CoAP Proxy-Uri option gives the absolute URI of the actual resource to be queried.

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 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 user 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.

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 HyperText Markup Language (HTML). This extension does not provide any benefit for M2M communication but is aimed towards humans. By rewriting the CoRE link format into HTML the information 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. For a more detailed description of our implementation, we refer to the study of Ishaq et al. (2012).

5 Next steps: basics for realising web of things

Using the above solution for self-configuration, newly deployed constrained devices are automatically discovered, their names are added to DNS and their resources are directly accessible and browsable over IPv6 via HTTP or CoAP or can be added to a directory server. This solution therefore presents an important building block that facilitates the actual usage of embedded web services as is required for building the WoT. Once end-to-end access to embedded web services has been realised, adding new functionalities or building novel services involving IoT objects is straightforward.

For example, the state of resources can be continuously observed using the CoAP observe extension (Hartke, 2012), leading to an as consistent as possible representation of resources. Using conditional observations (Teklemariam et al., 2013), interested parties can be notified about resource states that satisfy specific conditions, thereby acting as an enabler to build applications such as sensor – actuator interactions. These extensions enrich the capabilities of the basis CoAP protocol and contribute to the realisation of the WoT.

Looking at the resources themselves, several representations can be explored, ranging from plain text formats over formats defined by the IPSO (internet Protocol for Smart Objects) alliance (Shelby and Chauvenet, 2012) to complex semantic representations using ontologies that are adapted to the specific applications and domains as described for instance in the work of Abdulrazak et al. (2010). Also, the SPITFIRE project is providing vocabularies to describe sensors and to integrate them with the linked open

data cloud. Semantic descriptions of embedded web services allow linking sensor data with other available data. It brings the potential of semantic web technology (e.g. searching and reasoning) to constrained devices, realising a semantic WoT.

Further, through embedded web services, existing web service technologies, tools and frameworks become reusable for building web application and web services that are based on the state of the real world. However, it will impose novel challenges to web service aspects that are currently well understood, such as web service composition, due to the limitations of the constrained devices. For instance, embedded web services can be composed to create complex interaction scenarios where knowledge about the real world is used, linked with other services and processed to act again upon the physical world. Existing composition and orchestration frameworks as described by Yahyaoui et al. (2010) need to be extended in order to allow the incorporation of embedded web service technology and realise the WoT. Also, when time-varying data from constrained objects are incorporated or web services act upon the real-world issues such as consistency, failures, correct execution of all transactions as described by Gao et al. (2011) need to be explored in view of a constrained environment.

Using standardised technologies, powerful and scalable solutions for collecting, storing and processing a multitude of sensor data can be developed. The link with state-of-the-art cloud technology solutions that are gradually being adopted is clear (Kim, 2011). Also tiny objects can be introduced as part of grid computing, for example, for the collection and processing of environmental information. In the work of Rodriguez et al. (2011), an extensive overview of the introduction of mobile devices into grid systems is given and an extension to the constrained world seems feasible with the advent of embedded web service technology.

Similar to search engines in the world wide web, sensor resources could be indexed just as regular web pages and made available to internet users. Of course, issues such as time-dependent aspects should be taken into consideration (e.g. indexing a temperature sensor) introducing novel challenges and opportunities.

This short discussion clearly reveals the great potential offered through the availability of embedded web service technology. It can really facilitate the realisation of WoT services, opening up access to sensor data and stimulating their widespread usage, while at the same time avoiding vertically integrated and closed systems. As such, it presents great opportunities to researcher active as well in the field of web service technology as in the field of embedded distributed systems.

6 Deployment

Together, the mechanisms described in Section 4, realise 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 that includes both real and simulated sensor nodes. The implementation consists of two parts, the implementation running on the sensor nodes and the implementation running on the gateways.

The implementation on the gateways, called CoAP++, has been largely realised in click router, a C++ based modular framework that can be used to realise any network

packet processing functionality (Kohler et al., 2000). It consists of several modules such as the CoAP protocol, a CoAP server backend capable of offering resources, CoAP server discovery with DNS integration, HTTP-to-CoAP proxying and 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 realise the following configurations:

- A stand-alone socket-based CoAP client making use of IPv6/UDP sockets for network communication.
- 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 realisation of the gateway functionality.
- A CoAP sensor gateway with sensor discovery and DNS integration, and (non-) transparent proxy functionality. The sensor discovery module in the gateway discovers the sensors in the sensor network and, based on the subnet and domain info it has obtained statically or from the internet gateway, it creates the FQDN name and the mapping between IPv6 addresses and names. For example, if the sensor node with short address 1 and name node 1 is discovered by the sensor gateway and the sensor gateway is responsible for the sensor subnet aaaa::/64 and sensor domain subnet1.iot.test.ibbt.be, the FQDN name node1.subnet1.iot.test.ibbt.be is created and mapped to the IPv6 address aaaa::1.

  Next, this module interacts with dnsmasq, which runs locally on the gateway. First, as soon as the subnet and domain info are known, dnsmasq is configured in such a way that the sensor gateway acts as the forward and reverse DNS server for the sensor sub-domain and subnet. Next, when sensors are discovered and the FQDN-IPv6 mapping has been created; the service discovery module dynamically updates the dnsmasq configuration file, adding the forward and reverse DNS records, and signals dnsmasq about any changes. This way, it is made sure that the DNS server in the sensor gateway always has up-to-date information.
- A CoAP internet gateway with sensor gateway discovery and DNS integration and non-transparent proxy functionality. When the sensor gateway discovery module discovers sensor gateways the following will happen. If discovered sensor gateway already knows its subnet and domain, this information, together with the IPv6 address of the gateway, is used to update the dnsmasq configuration on the internet gateway. The sensor gateway is designated as the forward and reverse DNS server for the sensor subnet and all related DNS requests are forwarded to this gateway. Note that the internet gateway acts as the DNS server for all sensor subnets (which have a common domain). If the discovered sensor gateway does not know its subnet and domain, this information is dynamically assigned from a pool of domains and subnets. The information is sent back to the sensor gateway and dnsmasq on the internet gateway is updated as described before. Finally, in both cases, also a new routing entry is installed. As such, all incoming IPv6 packets can be routed to the correct sensor subnet.

The CoAP client/server protocol and the CoAP resources on the sensors have been implemented using the IDRA framework (De Poorter et al., 2011). IDRA is a network

architecture and application platform developed for TinyOS (http://www.tinyos.net/) and written in nesC (http://nescc.sourceforge.net/). The designed solution for the sensors has a footprint of 37,092 bytes in ROM and 5923 bytes in RAM. This footprint includes, in addition to CoAP, a simple (always on) Media Access Control (MAC) protocol and the Ad hoc On-Demand Distance Vector (AODV) routing protocol (Perkins et al., 2003). Such a small footprint can be run on the most-constrained devices.

During our testing we had debug info enabled; consuming part of the 48 KB ROM we had available. As such, we could only use the most basis MAC protocol, which is an always-on MAC. Thus we did not do an evaluation with a duty-cycled MAC protocol. In the future, it should be possible to further evaluate our solution in a real duty-cycled network when removing debug info or using new devices with, e.g., 92 KB ROM. Using another MAC will impact the performance of the solution. For example, when using an always-on MAC, the sender of a broadcast packet, only has to send the packet once, since every neighbouring node should be able receive it immediately. When using a duty-cycled MAC such as LPL, the sender has to send the packet repeatedly during the entire duty cycle, consuming more energy for transmission and increasing the chance of collisions. On the other hand, less energy will be consumed because the radio will be sleeping most of the time. Next to this, there are several other MACs such as synchronised MAC protocols. Also, specialised MAC protocols for energy-efficient broadcasting in constrained networks exist as well (Hurni and Braun, 2010). The performance impact of different MACs on the presented solution is outside the scope of this work, but is interesting to explore in the future.

Access to the testbed and its resources is possible to anyone that is connected to the internet using IPv6. The testbed can be accessed either over CoAP or over HTTP using the following URIs:

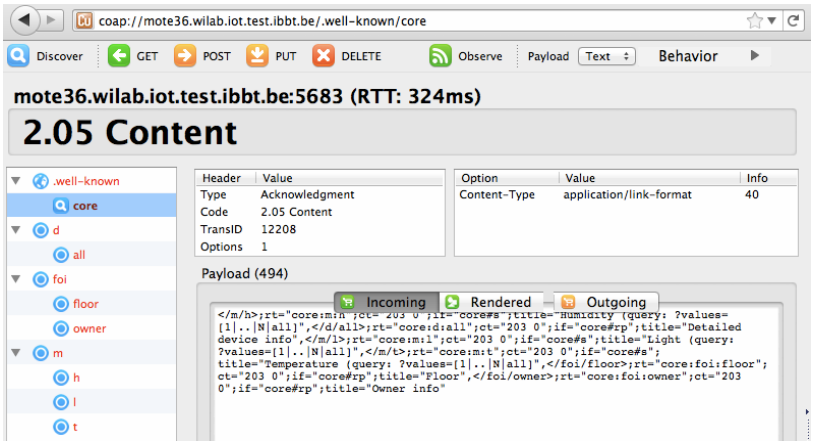
- coap://coap.iot.test.ibbt.be:5683/.well-known/core
- http://coap.iot.test.ibbt.be:8080/.well-known/core.

To illustrate the browsable hierarchy of the solution, a series of screenshots of HTTP access to the testbed is shown in Figure 5. For simplicity, the screenshots skip the level of the internet gateway and start at the level of a sensor gateway. With no prior knowledge about names, addresses or availability of the sensors and by using a standard web browser (without CoAP support) the client accessed the sensor gateway. In Figure 5a, the client was offered a list of available resources on the sensor gateway amongst which the client selected the ‘servers’ resource to get a list of the known sensors running CoAP (Figure 5b). The client selected one sensor ‘mote36’ and got back a list of the available resource on that sensor (Figure 5c). The client then selected the ‘m/t’ resource to get back the measured temperature on that sensor in a semantic description as shown in Figure 5d. In the entire process, the client was communicating using HTTP and HTML, while the gateway was translating to and from CoAP and core link format to be able to relay the communication to the sensors.

A sample of the direct end-to-end CoAP access to a resource on a sensor in the testbed is shown in Figure 6. The web browser used in this example was firefox (http://www.mozilla.org/firefox) with the copper (Cu) add-on (http://people.inf.ethz.ch/~mkovatsc/copper.php) installed in order to be able to handle CoAP. In this case, the sensor sent back the reply in CoRE link format as shown in Figure 6 (bottom right part). The add-on interpreted the CoRE link format and displayed it visually in the left side of Figure 6.

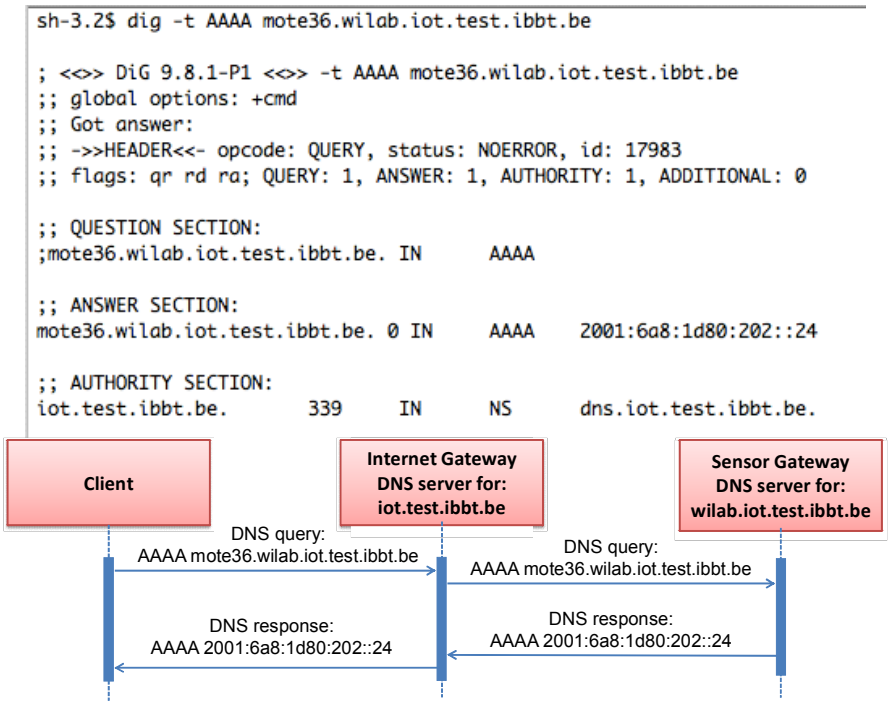
Figure 5 A step-by-step examples illustrating the browsable hierarchy as a client using a web browser might follow to discover and access sensor resources. (a) List of resource available on a sensor gateway, (b) list of discovered sensors running CoAP, (c) list of resources offered by a particular sensor, (d) the measured temperature on that sensor along its semantic description

Figure 6 A sample of a direct end-to-end access to a CoAP resource on a sensor in the testbed using a web browser with the copper add-on installed. On the bottom-right textbox the payload of the answer in CoRE link format is shown. On the left side a visual interpretation of this format is generated by the add-on



In the above examples, human friendly FQDN were used. As explained in Section 4, these names were automatically derived from the short sensor names and where automatically added to the relevant name servers. This is shown in Figure 7, where the discovered sensor information consisting of the name 'mote36' and 16-bit address '36' has been used to create an IPv6 address and an FQDN have been dynamically added to the DNS on the sensor gateway. When a client queried the internet gateway for the IPv6 for that sensor, the internet gateway forwarded the query to the appropriate sensor gateway, got a response from it and relayed the response back to the client.

Figure 7 Name resolution using dynamically configured DNS information. The client queries for a sensor address (mote36.wilab.iot.test.ibbt.be). The request is received by the internet gateway, which is the DNS server for iot.test.ibbt.be and forwards it to the sensor gateway since it is the DNS server for the sub-domain wilab.iot.test.ibbt.be. The response consisting of the dynamically created IPv6 address from the short sensor address (36) is sent back to the client (see online version for colours)



7 Analysis of the different approaches

As mentioned in Section 4.2, we have implemented different approaches for the discovery of sensors inside the sensor network using CoAP. The sensor gateway can search for sensors (pull approach) or the sensors can announce their presence to the gateway (push approach). To validate these approaches in sensor discovery we conducted several experiments using a real-life wireless sensor network testbed, namely w-iLab.t

(Bouckaert et al., 2010). This testbed is deployed across an operational office building with a significant number of co-located company wireless devices (Wi-Fi routers, DECT phones, Bluetooth devices, etc.) that cause interference to the sensors in the testbed and make the test networks lossy as the case in realistic environments. The experiments were run on two sensor networks inside the testbed with different network characteristics. The first sensor test network had 24 sensor nodes in a topology with a maximum broadcast domain of 22 nodes and nodes were within a maximum of two hops from the sensor gateway. The second test sensor network had 52 sensor nodes in a topology with a maximum broadcast domain of 38 nodes and nodes were within a maximum of two hops from the sensor gateway. In the following subsections, we present the results of these experiments.

7.1 Pull approach

The advantage of the pull approach in discovering the sensors is that it can be manually executed whenever needed, for example, when the network is built or whenever it is modified. Manually triggered discovery can be complemented with periodic pulls to discover changes in the network and possibly generate alerts to the administrator. However the pull approach relies on sending multicasts, which are often translated into broadcasts inside sensor networks. Depending on the size and the topology of the network and on how broadcasts are forwarded into the network, frequent broadcasts can easily lead to network congestion. Thus, periodic pulls should be kept to a minimum and at relatively large intervals in order to avoid network congestion. Due to these potential congestions and to the fact that CoAP uses non-confirmable messages to reply to broadcasts, it is anticipated that some sensors might not get discovered immediately after the first discovery broadcast.

As anticipated, our tests revealed that using broadcasts as a means to discover the available sensors is challenging. In fact already with a broadcast domain of 22 nodes as was the case in our smaller test network, no sensors were initially discovered. The reason is that the sensors crashed as a result of sending just one discovery broadcast by the sensor gateway. This single broadcast packet was received by most sensors in the network and triggered those sensors to respond to it almost at the same time. Since the route to the destination of the response (the sensor gateway) was unknown to most sensors, the routing protocol on these sensors (AODV in our case) also started broadcasting to find routes to the sensor gateway. This ultimately led to a broadcast storm in the network and a receive buffer overflow in most sensors in the test network.

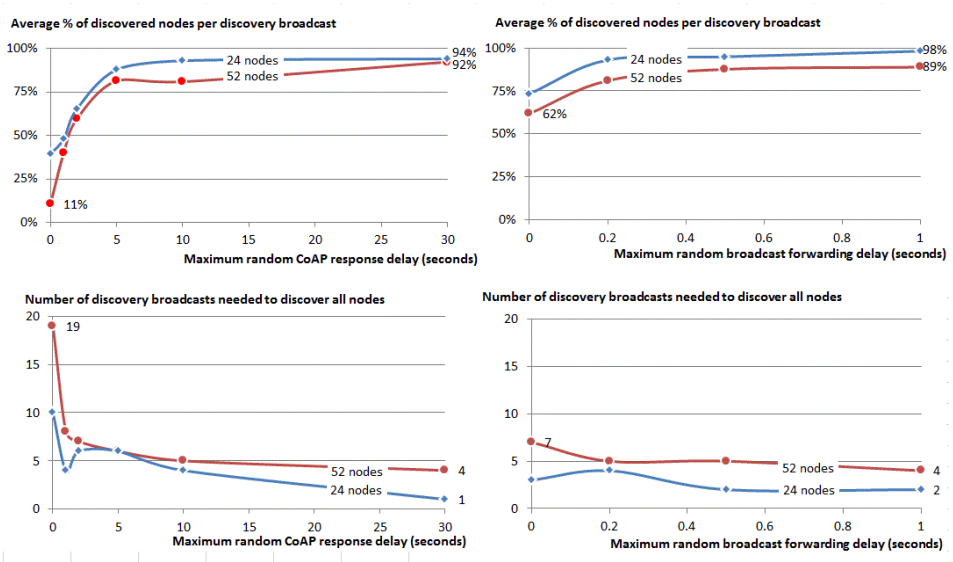
However in order to combat broadcast storms in the network and to improve the discovery rate, three techniques were used and examined. The first technique was to let the sensor gateway broadcast a route reply packet in the network. This caused all sensors to establish a route to the gateway and eliminated the need for them to start asking for a route once they get the discovery request. This way measuring the overhead of calculating routes to the gateway is avoided, since the evaluation of the used routing protocol is beyond the scope of this work. The other two techniques introduced random delays in the network – before responding the CoAP discovery requests and before forwarding broadcasts in the network.

The effect of introducing random delays before sensors respond to discovery broadcasts on the percentage of discovered sensors for the two test networks is shown in Figure 8 (top left corner). In CoAP terms, this delay is called leisure. The server could

either use a default value for leisure or compute a value for it. If the server has a group size estimate G, a target data transfer rate R and an estimated response size S, a rough lower bound for leisure can then be computed as

$$Leisure_{lowerbound} = \frac{S * G}{R}$$

Figure 8 The effect of introducing random delays before sensors respond to CoAP discovery broadcasts and before they forward broadcasts to other sensors for two different network sizes. By randomly delaying the responds for a few seconds and the broadcast forwarding a few hundred milliseconds the discovery rate is highly improved (see online version for colours)



For our two test networks G equals 22 and 38, S equals 100 bytes, and the target rate can be set to a conservative 8 Kb/s = 1 KB/s. The resulting lower bound for the leisure is then equal to 2.2–3.8 s, respectively. For a more complete discussion of the leisure period and its estimation we refer to Section 8.2 of Shelby et al. (2012).

The graph shows that the average percentage of discovered nodes decreases with increase of broadcast domains. The averages were computed for sending at least ten broadcasts for each experiment setting. The graph also shows that already by randomly delaying the responses for a few seconds the discovery rate was highly improved. However it also shows that sending only one broadcast discovery CoAP request does not guarantee the discovery of all sensors in the network. This is due to the lossy nature of sensor networks and to the fact that broadcasts are not acknowledged and thus some sensors might not receive it. In Figure 8 (the bottom left corner) the number of needed broadcasts to discover all sensors in the network is shown for the above experiments. The graph shows that by the introduction of a random delay of a few seconds all sensors are discovered significantly faster. Please note that these experiments were taken while having a maximum random rebroadcast delay of 200 ms (see next paragraph).

In Figure 8 (the right side) the effect of introducing a random delay before sensors forward any received broadcast to the other sensors on the discovery rate and on the number of needed requests to discover all sensors is shown. The graphs show that by delaying the rebroadcasting for several hundred milliseconds, the discovery rate is highly improved and thus fewer requests are needed to discover all sensors. Please note that these measurements were taken while having leisure of 10 s as explained in the previous paragraph.

Table 1 summarises the results of the various experiments of the discovery following the pull approach and additionally shows the average discovery delays per sensor. Such delays are in most use cases for sensor discovery irrelevant and can be tolerated. Please note that all these measurements were taken after all sensors have established a route as result of the broadcast route reply mentioned earlier in this section.

Table 1 Summary of the experiments showing the effect of changing the maximum random delays in forwarding broadcasts and in responding to CoAP discovery broadcasts (leisure) on the discovery rate and delay for two different test networks. The averages were computed for sending at least ten broadcasts for each experimental setting

Broadcast forwarding delay [s]	*Leisure [s]*	*Network size [# of nodes]*	*Average discovery delay [s]*	*Average discovered nodes per request [%]*	*# of BC to discover all nodes*
0	0	24	Sensors crash due to broadcast storms		
		52			
	10	24	5.80	73%	3
		52	5.58	62%	7
0.2	0	24	1.69	39%	10
		52	1.08	11%	19
	1	24	1.87	48%	4
		52	3.00	40%	8
	2	24	2.31	65%	6
		52	3.51	60%	7
	5	24	3.31	88%	6
		52	4.46	81%	6
	10	24	5.72	93%	5
		52	5.62	81%	4
	30	24	13.98	94%	1
		52	12.68	92%	4
0.5	10	24	5.93	95%	2
		52	5.92	88%	5
1	10	24	5.91	86%	2
		52	5.45	89%	4

7.2 Push approach

As an alternative to the pull approach described in the previous subsection, it is possible to configure the sensors to announce their presence to the sensor gateway (push approach). To realise this, the sensors are hardcoded with an anycast address to which they send their announcement messages. This anycast address identifies a group of

potential receivers. An anycast packet is routed in a similar way as a unicast packet until it reaches a receiver of the group. This first receiver will process the packet and stop it forwarding. By configuring the gateway to listen to this anycast address, the gateway is able to receive all announcement messages. This approach has the advantage that the address of the sensor gateway does not have to be known by the sensors.

As mentioned in Section 2, reliability of CoAP messages is optional. The sender can elect to either use confirmable or non-confirmable messages. By using confirmable messages, reliable delivery of the messages is guaranteed. Using non-confirmable messages reduces network traffic by eliminating the need for transmitting acknowledgments but does not guarantee that the messages are really delivered.

Our tests revealed that when using non-confirmable CoAP anycast messages the sensor discovery rate and thus the number of transmissions needed to make sure that the gateway has indeed discovered, all sensors were very similar to using the pull approach. This is not surprising, since in both cases delivery of the messages depends on the load on the sensors and the network. When varying the random start-up delay of the nodes from 5 ms to 5 s, we observed discovery percentages increasing from 79% to 100%. A very low start-up delay means that most nodes will simultaneously announce their presence to the gateway, resulting in collisions. In this case, confirmable messages can help to alleviate this problem. However, it is expected that in real-life situations, nodes will not start up almost simultaneously.

This push-based sending of anycast messages can be done once at boot time or periodically. Obviously when done only at boot time, there is a risk that the sensor gateway was not running or missed the registration and thus the sensor remains undiscovered. Periodically repeating the push-based sending of anycast messages is possible, but the frequency should be limited in order not to consume too much energy for sending and forwarding these messages. Of course, there will be a straightforward trade-off between energy consumption and speed of discovery. However this approach still makes sense to use when only a few sensors have been added to the network.

Another issue that should be considered when implementing the push approach on constrained devices is that the implementation will be slightly more complex and potentially have a higher footprint since it involves the use of timers in the code.

7.3 Pull-push approach

The two discovery approaches can also be combined together as needed. For example, one can use the pull approach for initial discovery of the network and for periodical re-discovery at relatively large time intervals. The administrator can also trigger a pull-based re-discovery when he/she thinks there are changes in the network. At the same time, push-based notifications can be used so that sensors can announce their presence in the network, without waiting for the next pull cycle.

8 Comparison with related work

8.1 Related work

In this section, we discuss related work focusing on the automatic discovery of sensors, realisation 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, Cosm (http://www.cosm.com) 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 (Reed et al., 2007) defines web service interfaces for accessing sensor data, controlling sensors and alerting, functionalities comparable to the ones offered by CoAP. Jirka et al. (2009) present the OSIRIS sensor web discovery framework, which makes use of registries that are being built and which are capable of handling the dynamic properties of sensors. Similarly, a web crawler could periodically scan the WoT for sensors and downloading metadata 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 (Pfisterer et al., 2011).

These solutions aim for the realisation of a semantic WoT or sensor web, which enables data producers and users to publish and access sensor information via web- and standard-based interfaces (see Foerster et al., 2012, 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 network are not considered, although very important. Our solution provides an answer to these problems, which can be seen as a building block for the realisation of the semantic WoT and is therefore quite different, but complementary.

When focusing more on the networking aspect (discovery followed by integration in DNS and automatic routing to sensor subnets), almost no related work is found. Östmark et al. (2006) present a solution 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-organisation 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. Schneider et al. (2011) present a solution 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, Schor et al. (2009) describe a zero-configuration IPv6/6LoWPAN-based system architecture. 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-organising hierarchical levels automatically linked

with each other, resulting in a browsable discovery system that allows the discovery 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.

8.2 High-level comparison

As already mentioned, the deployment of sensor networks, including their integration in the internet, is a multi-faceted problem consisting of hardware provisioning, actual hardware installation and placement, optional calibration, the creation of an operational sensor network, *connectivity with existing networks, discovery (user-friendly) access*, management and maintenance of the operational network. The solution proposed in this paper aims to facilitate and automate the steps marked in italic. For example, automatic DNS integration of sensors and sensors gateways and automatic routing towards the sensor network saves the installer time otherwise spent on manual configurations. After the discovery, an installer can easily interact with the discovered sensors in a RESTful way for further configuration, benefitting from features such as the DNS integration, HTTP/CoAP proxying and HTML translation.

As described in Section 8.1, there exist other solutions that fulfil part of the functionality required during deployment and the life cycle of a sensor network. When using other solutions, an installer will still have to go through all the different steps inherent to any deployment scenario. Compared to solutions we are aware of, our solution offers improvements in terms of automation: e.g. automatic DNS integration, automatic routing towards the sensor network. In addition, it offers improvements in terms of footprint. Only a CoAP implementation is needed and no different protocols are being used for both discovery and resource access. As such, our solution nicely aligns with ongoing standardisation efforts for constrained networks in IETF and its footprint fits on the most constrained devices. This is different from, for example, solutions based on HTTP (which has too much overhead for a constrained network), mDNS (which requires additional protocols for resource access) or solutions coming from closed standardisation bodies such as the Zigbee alliance.

9 Conclusions

In this paper, we have described a novel self-organisation 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-to-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. At this point, existing web crawler solutions can be used to index and publish the sensor information to the outside world. As such, the proposed approach provides a feasible and flexible solution to achieve hierarchical self-organisation 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 and

access to sensor networks. As such, it represents an important building block facilitating the actual usage of embedded web services as is required for building the WoT. Once end-to-end access has been realised, adding new functionalities or building novel services involving IoT objects is straightforward and many opportunities to integrate this with existing web service technologies arise.

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 set-up. This evaluation gave us insights about the strengths and limitations of the proposed approach in a large-scale, real-life environment. In the future, the solution will be further refined and tested, extended with other CoRE functionality such as caching 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-organisation, configuration and management of sensor networks to a next level.

Acknowledgements

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 258885 (SPITFIRE project), from the iMinds ICON project O'CareCloudS, from a VLIR PhD grant to Isam Ishaq and through an FWO postdoc research grant for Eli De Poorter.

References

Abdulrazak, B., Chikhaoui, B., Vallerand, C.G. and Fraikin, B. (2010) 'A standard ontology for smart spaces', *International Journal of Web and Grid Services*, Vol. 6, No. 3, pp.244–268.

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

Bormann, C. (2012) *Guidance for Light-Weight Implementations of the Internet Protocol Suite draft-ietf-lwig-guidance-02*, Internet-Draft, IETF.

Castellani, A.P., Loreto, S., Rahman, A., Fossati, T. and Dijk, E. (2012) *Best Practices for HTTP-CoAP Mapping Implementation draft-castellani-core-http-mapping-05*, IETF.

Chen, M., Mao, S., Xiao, Y., Li, M. and Leung, V.C.M. (2009) 'IPSA: a novel architecture design for integrating IP and sensor networks', *International Journal of Sensor Networks*, Vol. 5, No. 1, pp.48–57.

Colitti, W., Steenhaut, K. and Caro, N. De. (2011a) 'Integrating wireless sensor networks with the web', *Proceedings of Workshop on Extending the Internet to Low power and Lossy Networks*, 11 April, Chicago, Illinois.

Colitti, W., Steenhaut, K., De Caro, N., Buta, B. and Dobrota, V. (2011b) 'Evaluation of constrained application protocol for wireless sensor networks', *18th IEEE Workshop on Local & Metropolitan Area Networks (LANMAN)*, IEEE, pp.1–6, doi:10.1109/LANMAN.2011.6076934.

De Poorter, E., Troubleyn, E., Moerman, I. and Demeester, P. (2011) 'IDRA: a flexible system architecture for next generation wireless sensor networks', *Wireless Networks*, Vol. 17, No. 6, pp.1423–1440.

Duquennoy, S., Wirström, N., Tsiftes, N. and Dunkels, A. (2011) 'Leveraging IP for sensor network deployment', *Proceedings of the Workshop on Extending the Internet to Low Power and Lossy Networks (IP+SN 2011)*, Chicago, Illinois.

Foerster, T., Nüst, D., Bröring, A. and Jirka, S. (2012) 'Discovering the sensor web through mobile applications', Gartner, G. and Ortag, F. (Eds): *Advances in Location-Based Services*, Springer, Berlin Heidelberg, doi:10.1007/978-3-642-24198-7.

Gao, L., Urban, S.D. and Ramachandran, J. (2011) 'A survey of transactional issues for web service composition and recovery', *International Journal of Web and Grid Services*, Vol. 7, No. 4, pp.331–356.

Hartke, K. (2012) *Observing Resources in CoAP draft-ietf-core-observe-06*, IETF.

Hui, J.W. and Culler, D.E. (2008) 'IP is dead, long live IP for wireless sensor networks', *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems – SenSys'08*, ACM Press, New York, USA, doi:10.1145/1460412.1460415.

Hurni, P. and Braun, T. (2010) 'An energy-efficient broadcasting scheme for unsynchronized wireless sensor MAC protocols', *7th International Conference on Wireless On-demand Network Systems and Services (WONS)*, IEEE, pp.39–46, doi:10.1109/WONS.2010.5437134.

Ishaq, I., Carels, D., Teklemariam, G., Hoebeke, J., Abeele, F., Poorter, E. and Demeester, P. (2013) 'IETF standardization in the field of the internet of things (IoT): a survey', *Journal of Sensor and Actuator Networks*, Vol. 2, No. 2, pp.235–287.

Ishaq, I., Hoebeke, J., Rossey, J., De Poorter, E., Moerman, I. and Demeester, P. (2012) 'Facilitating sensor deployment, discovery and resource access using embedded web services', *6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, IEEE, pp.717–724, doi:10.1109/IMIS.2012.48.

Jirka, S., Bröring, A. and Stasch, C. (2009) 'Discovery mechanisms for the sensor web', *Sensors (Basel, Switzerland)*, Vol. 9, No. 4, pp.2661–2681.

Kim, W. (2011) 'Cloud computing adoption', *International Journal of Web and Grid Services*, Vol. 7, No. 3, pp.225–245.

Kohler, E., Morris, R., Chen, B., Jannotti, J. and Kaashoek, M.F. (2000) 'The click modular router', *ACM Transactions on Computer Systems*, Vol. 18, No. 3, pp.263–297, doi:10.1145/354871.354874.

Kuladinithi, K., Bergmann, O., Pötsch, T., Becker, M. and Görg, C. (2011) 'Implementation of CoAP and its application in transport logistics', *Extending the Internet to Low power and Lossy Networks' (IP+SN 2011)*, Chicago, USA.

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

Montenegro, G., Kushalnagar, N., Hui, J.W. and Culler, D.E. (2007) *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*, IETF.

Östmark, Å., Eliasson, J., Lindgren, P., Meppelink, L. and Van Halteren, A. (2006) 'An infrastructure for service oriented sensor networks', *Journal of Computers*, Vol. 1, No. 5, pp.20–29, doi:10.4304/jcp.1.5.20-29.

Perkins, C.E., Belding-Royer, E.M. and Das, S.R. (2003) *Ad hoc On-Demand Distance Vector (AODV) Routing*, IETF.

Pfisterer, D., Römer, K., Bimschas, D., Kleine, O., Mietz, R. and Truong, C. (2011) 'SPITFIRE: toward a semantic web of things', *IEEE Communications Magazine*, November, pp.40–48.

Pötsch, T. (2011) *Performance of the Constrained Application Protocol for Wireless Sensor Networks*. Available online at: http://www.comnets.uni-bremen.de/itg/itgfg521/aktuelles/fg-workshop-29092011/ITG_HH_thomas_poetsch.pdf (accessed on 29 December 2012).

Reed, C., Botts, M. and Davidson, J. (2007) 'OGC® sensor web enablement: overview and high level architecture', *IEEE Autotestcon*, IEEE, pp.372–380, doi:10.1109/AUTEST.2007.4374243.

Rodriguez, J.M., Zunino, A. and Campo, M. (2011) 'Introducing mobile devices into grid systems: a survey', *International Journal of Web and Grid Services*, Vol. 7, No. 1, pp.1–40, doi:10.1504/IJWGS.2011.038386.

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

Schor, L., Sommer, P. and Wattenhofer, R. (2009) 'Towards a zero-configuration wireless sensor network architecture for smart buildings', *Proceedings of the 1st ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings – BuildSys'09*, ACM Press, New York, New York, USA, p. 31, doi:10.1145/1810279.1810287.

Shelby, Z. (2010) 'Embedded web services', *Wireless Communications, IEEE*, Vol. 291, No. 4, pp.76–81, doi:10.1109/MWC.2010.5675778.

Shelby, Z. (2012) *Constrained RESTful Environments (CoRE) Link Format, draft-ietf-core-link-format-11*, IETF, Internet-draft.

Shelby, Z. and Bormann, C. (2009) *6LoWPAN – The Wireless Embedded Internet*, Wiley.

Shelby, Z. and Chauvenet, C. (2012) *The IPSO Application Framework draft-ipso-app-Framework-04*, IPSO Alliance.

Shelby, Z., Hartke, K., Bormann, C. and Frank, B. (2012) *Constrained Application Protocol (CoAP) draft-ietf-core-coap-12*, IETF, Internet-draft.

Teklemariam, G., Hoebeke, J., Moerman, I. and Demeester, P. (2013) 'Facilitating the creation of IoT applications through conditional observations in CoAP', *EURASIP Journal on Wireless Communications and Networking*, Vol. 2013, No. 1, pp.177, doi:10.1186/1687-1499-2013-177.

Vasseur, J-P. and Dunkels, A. (2010) *Interconnecting Smart Objects with IP: The Next Internet*, Morgan Kaufmann, San Francisco, CA.

IoT-A (2011) *White Paper: Smart Networked Objects & Internet of Things*, News.

Winter, T., Thubert, P., Brandt, A., Hui, J.W., Kelsey, R., Levis, P. and Alexander, R.K. (Eds.) (2012). *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*, IETF.

Yahyaoui, H., Maamar, Z. and Boukadi, K. (2010) 'A framework to coordinate web services in composition scenarios', *International Journal of Web and Grid Services*, Vol. 6, No. 2, pp.95–123, doi:10.1504/IJWGS.2010.033787.

Yazar, D. and Dunkels, A. (2009) 'Efficient application integration in IP-based sensor networks', *Proceedings of the 1st ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings – BuildSys'09*, ACM Press, New York, New York, USA, p.43, doi:10.1145/1810279.1810289.

Zheng, J. and Jamalipour, A. (2009) *Wireless Sensor Networks – A Networking Perspective*, Wiley.

ZigBee IP. (n.d.) *ZigBee IP Specification Overview*. Available online at: http://www.zigbee.org/Specifications/ZigBeeIP/Overview.aspx (accessed on 24 April 2013).