Demo Abstract: Building embedded applications via REST services for the Internet of Things

Floris Van den Abeele, Jeroen Hoebeke, Isam Ishaq, Girum K Teklemariam, Jen Rossey, Ingrid Moerman, Piet Demeester
Department of Information Technology (INTEC)
Ghent University – iMinds
{fvdabeele,jhoebeke,iishaq,gketema,jrossey,imoerman,pietdm}@intec.ugent.be

ABSTRACT

As embedded networks are evolving to open systems, it's becoming possible to create new applications on top of these existing embedded systems. However, developing new applications can be difficult due to the large diversity of protocols that exist today. In this paper, the authors demonstrate how employing the CoAP protocol can enable rapid application development by re-using well-known principles from the Web development world. Furthermore, we also demonstrate how a number of extensions to CoAP help to lower the barrier for developing applications even further.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*

1. INTRODUCTION

In recent years we've seen how embedded networks have been evolving from closed, self-contained systems to open and accessible systems. In the past, an embedded deployment would typically be a tightly vertically integrated structure. This tight integration meant that re-using such networks for applications different than the one it was originally designed for, was often difficult and time consuming. Today however, a lot of these embedded networks have opened up their doors. One of the key drivers for this evolution has been the ongoing adoption of open standards. For instance, the IETF 6LoWPAN working group has defined an adaption layer [1] for IEEE 802.15.4 networks (a popular networking technology for embedded networks) that has allowed embedded devices to become 1st class citizens of the IPv6 Internet. In doing so, 6LoWPAN has made end-to-end IPv6 connectivity possible with any other IPv6 host.

In a next step, the IETF looked at standardizing a protocol for exchanging information with constrained devices. Thus, the Constrained RESTful Environments (CoRE) working group and the CoAP [2] protocol were born. CoAP proposes to employ a Web like REST architecture to structure embedded applications. Applications are structured by splitting them up into REST resources that reside on the embedded device. Communication follows a simple request-response model where clients query REST resources to retrieve information or to control the device and its physical environment. This makes creating applications for embedded devices very similar to Web service development.

The authors combine the CoAP protocol with a number of extensions to demonstrate how one can easily create new applications on top of embedded networks.

SenSys'13, Nov 11–15, 2013, Roma, Italy.
ACM 978-1-4503-2027-6/13/11
http://dx.doi.org/10.1145/2517351.2517426 ...$15.00.

2. EXTENSIONS ON TOP OF COAP

A lot of extensions that run on top of the CoAP protocol are currently under development. As some of these will be demonstrated in this demo, a short overview is presented here. One of these extensions, "Conditional observe in CoAP" [3], extends the publish/subscribe mechanism from CoAP with server-side filtering of notifications. Instead of having to subscribe for all state changes of a resource and applying client-side filtering, a client can instead subscribe for specific events in which it is interested. Another extension that will be demonstrated is the use of a messaging (de)multiplexer, as defined in the CoAP Entities [4] Internet draft. This so-called "Entity Manager" allows clients to access groups of embedded devices even when multicast IP is not supported in the embedded network. Finally, the "CoRE Resource Directory" [5] is a REST service where embedded devices and their resources can be registered and where an application can search for resources.

3. SYSTEM OVERVIEW

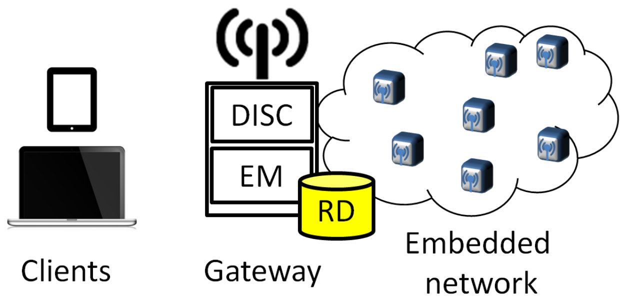

Figure 1: Overview of components

Figure 1 gives an overview of the different components that are used in the demonstration. The network pictured on the right consists of 7 embedded devices in the form of Zolertia Z1's boards that are running the Contiki operating system. Each of these devices has been equipped with a number of sensors and actuators. The sensors include light intensity, temperature, proximity, movement (PIR), force, RFID and magnetic switch sensors. Supported actuators include multiple lights (in the form of LEDs) and a cooling fan. These sensors and actuators each have a corresponding REST resource, which implement the conditional observe extension if useful. The gateway is shown in the center of the figure and runs on an Alix system board. Apart from routing traffic and applying 6LoWPAN, the gateway also provides three other services: a discovery process (DISC), the entity manager (EM) and the resource directory (RD). Each of these services has been implemented in our modular CoAP framework in Click Router. The discovery process is in essence a periodic poll from the gateway to all devices in the embedded network, the specifics of which are described in [6]. The personal computer client on the bottom-left is only running the client module of our CoAP framework combined with a GUI that implements and represents the scenarios from section 4. The tablet client in the top-left is a non-native CoAP device that can interact with the embedded devices via a HTTP/CoAP cross-protocol proxy that is running on the gateway (proxy not shown in figure).

4. DEMONSTRATION

We have developed a total of six home automation scenarios on top of a portable embedded network that illustrate how the presented extensions can be used to ease the creation of applications. To illustrate how each of the extensions is used, two of the six scenarios are explained in greater detail; the other four are only briefly mentioned.

The first scenario occurs when a new device is deployed in the embedded network. After joining the network, the device and its resources are discovered by the gateway. The gateway subsequently adds the device and its resources to the resource directory. Clients querying the directory will notice that the directory has been updated and can build applications on top of the discovered resources. The other scenarios actually use this information to build their applications. So, in all other scenarios we assume that this first scenario has already been completed.

The conditional observe and entity extensions are illustrated in this second scenario. Here, the application is to switch on the lights whenever the light intensity drops below 500 Lux. The client (C) starts by grouping all the lighting actuator resources (A,B,C) into a "/lights" entity on the gateway (GW). This is done via a CoAP POST request to the /em resource provided by the entity manager. In a second configuration step, the client subscribes to the light intensity resource (S), asking to be notified whenever the measured value drops below 500 Lux. After completing the configuration, the light intensity sensor is covered with a dark cloth. This will trigger a notification from the light intensity sensor, in response to which the client is configured to send a CoAP POST request to the resource /light residing on the entity manager to turn on the lights. The entity manager takes cares of sending a request to each of the individual lighting actuators and replies with an aggregated response to the client when it has received a confirmation from every lighting actuator in the entity. The COAP URIs corresponding to A,B,C and S have been determined as the result of scenario 1. The scenario is illustrated in figure 2.

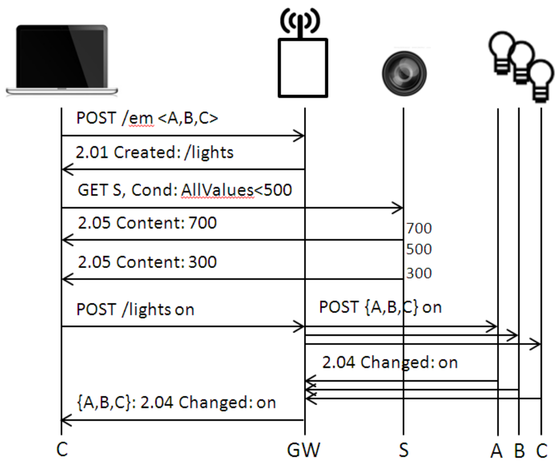

Figure 2: Interactions in scenario 2

The other four scenarios are similar to the previous one and are briefly listed here. In scenario 3, a rise of temperature activates the fan actuator. In the fourth scenario, the arrival of a miniature car opens the door (represented by a green LED) and turns on the lighting if it is dark. In the fifth scenario, the client is configured to alert the user via text and/or twitter when movement is detected by the PIR sensor. In the last scenario, the door is opened when a (configurable) tag is read by the RFID reader.

5. ACKNOWLEDGMENTS

Part of 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). This research is also partially funded by the O'CareCloudS project, a project cofunded by iMinds (Interdisciplinary Institute for Technology). iMinds is a research institute founded by the Flemish Government.

6. REFERENCES

[1] J. Hui and P. Thubert. Rfc 6282: Compression format for ipv6 datagrams over ieee 802.15.4-based networks.

[2] Z. Shelby, K. Hartke, and C. Bormann. Constrained application protocol (coap) (ietf internet draft v18).

[3] Girum Teklemariam, Jeroen Hoebeke, Ingrid Moerman, and Piet Demeester. Facilitating the creation of iot applications through conditional observations in coap. *EURASIP Journal on Wireless Communications and Networking*, 2013(1):177, 2013.

[4] I. Ishaq, J. Hoebeke, and F. Van den Abeele. Coap entities (ietf internet draft v0)).

[5] Z. Shelby, S. Krco, and C. Bormann. Core resource directory (ietf internet draft v5)).

[6] I. Ishaq, J. Hoebeke, J. Rossey, E. De Poorter, I. Moerman, and P. Demeester. Facilitating sensor deployment, discovery and resource access using embedded web services. In *Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, 2012.