Building Web-based Infrastructures for Smart Meters

Andreas Kamilaris[1], Vlad Trifa[2], and Dominique Guinard[2]

[1]University of Cyprus, Nicosia, Cyprus
[2]ETH Zurich and SAP Research, Switzerland

Abstract. *Smart Meters have been massively deployed recently, in order to provide energy awareness to people, helping them reduce their electricity footprint. We propose a Web-based infrastructure for integrating Smart Meters in future houses, providing high interoperability and scalability. We show that, by reusing the core principles of the modern Web architecture, we can build flexible applications on top of heterogeneous Smart Meters with little effort and acceptable performance.*

1 Introduction

Sustainability and energy conservation has become a major challenge in the world today with tremendous economic, political, and environmental implications. Increasing awareness about our energy consumption is an important factor in order to overall reduce electrical consumption. According to Google[1], consumers can save 5-15% on their electricity usage when they can visualize their energy consumption in real time.

In the latest years, Smart Meters have been gaining a certain popularity. Smart Meters, are wireless devices that measure in real-time the energy consumption of various electrical devices and control their operation. It is planned that every home in Britain will be equipped with Smart Meters by the end of 2020[2].

Traditional Smart Meters offer a house-level granularity, where only the whole house energy consumption can be visualized. As the technology becomes more advanced, monitoring the energy consumption of each electrical appliance becomes possible. However, a common standard for collecting energy consumption from several devices, manufactured by various constructors, is virtually inexistent. To void this gap, we propose here a simple infrastructure for connecting heterogeneous Smart Meters, which leverages the existing Web infrastructure. This infrastructure builds upon our previous work in building the *Web of Things* [2]. The core idea is to reuse the popular and wide-spread Web standards to interconnect embedded devices, and apply these principles to build an programmable ecosystem of Smart Meters. This will facilitate significantly the prototyping and development of applications that can contribute to energy reduction by raising the awareness about our own energy by releasing this information into the Web.

[1] `http://www.google.org/powermeter/sgtestimony.html`
[2] `http://news.bbc.co.uk/2/hi/business/8042716.stm`

2 Web-enabling Smart Meters

We propose a Web-oriented approach that enables Smart Meters to speak the same language as any other resource on the Web. This means we utilize HTTP to develop an open environment of hardware and software for energy monitoring. The rationale of using the Web as application layer is its many features that would be adapted to such a task, in particular its ubiquity and scalability. This is done by using the REpresentational State Transfer (REST) [1] as a set of constraints to make Smart Meters an integral part of the Web. REST is the architectural style behind HTTP and advocates in providing Web Services and data modeled as *resources*, unambiguously identified by unique resource identifiers (URI). Resources can be manipulated through the four common verbs specified in the HTTP standard: *GET* is used to retrieve a representation of a resource, *POST* alters the state of resources, *PUT* represents an insert or update, and *DELETE* is used to remove resources. Web services are a viable mechanism for use in embedded devices and Smart Building deployments [3, 4].

Hereafter, we identify the challenges to consider, for Web-enabling Smart Meters. Our work relies on the RESTful gateways described in [2], to bridge proprietary Smart Meters with the Web. Gateways can be bypassed by embedding a RESTful Web server directly on the Smart Meters, but where the Web-enablement takes place has no influence on the system proposed here, as it is fully transparent thanks to the properties of the HTTP protocol.

2.1 Discovery

HTTP does not have a mechanism for device discovery, because discovery in REST is done by following links. Therefore, we propose a simple process for discovering RESTful Smart Meters that aren't linked. Fig. 1 shows the general message interaction pattern followed at the discovery procedure.

When the Smart Meter is powered on, it will broadcast periodically a HELLO message (UDP broadcast on the local network if connected through Ethernet, or radio broadcast if the meter uses Bluetooth or ZigBee). When a gateway receives this HELLO message, it will acknowledge it and "bind" the gateway with the Smart Meter that generated it. Then, the meter responds with a message that contains the device description information and/or a description URL. The URL points to a Web page where a description of the resources offered by that particular meter can be found (can be either on the device itself or on the manufacturer's Website). The gateway receives this message, parses the contents of the URL and exposes the functions offered by the Smart Meter on the Internet.

2.2 Description

As mentioned earlier, the URL sent by the Smart Meter points to a page that contains a machine-readable description of the resources offered by the device. These resources can be the current electricity consumption measured (Watts), total consumption over a time range (kWh), remote control of electrical appliances

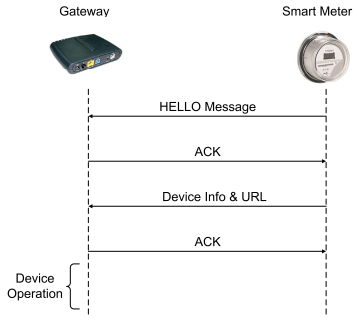

Fig. 1. Smart Meter Discovery Procedure.

(switch on/off the device attached to the meter), etc. To achieve interoperability with heterogeneous devices, we emphasized on a standard resource description language, and opted for Web Application Description Language (WADL[3]). WADL is an XML-based file format that provides a machine-readable description of HTTP-based web applications, particularly useful for describing RESTful Web Services.

2.3 Eventing

Energy measurements from Smart Meters can be filtered according to their importance and natures. These events can indicate simple occurrences such as turning on/off of an electrical appliance or more urgent incidents such as electricity leakage or fire. Gateways can constitute the backbone of a scalable, resource-oriented eventing infrastructure to efficiently disseminate events to interested entities. Notifications rely topic-based publish/subscribe mechanisms through Web *push* techniques. Any computing device that runs a Web server can be a subscriber that is notified through POST requests. This technique is called *Web Hooks*[4], which are event notifications via HTTP callbacks.

2.4 Data Collection and Exploitation

Gathering data produced by Smart Meters into a central location is necessary to transform it into useful information about electricity usage patterns of people, neighborhoods, cities or even countries. Open access to this information is a key enabler to increase awareness about our own energy usage. Home occupants will be able to compare their current electricity footprint over months and correlate their behavior with energy and money savings. Enabling direct access to this data through a Web-based RESTful API, will make very simple to integrate this data with existing Web applications, in particular social networks such as Facebook and Twitter to further involve users into sharing and comparing their energy consumption with their friends and relatives.

[3] https://wadl.dev.java.net/
[4] http://www.webhooks.org/

3 Implementation

Our RESTful gateway is composed of four principal layers: the *device layer* is responsible for the discovery and control of Smart Meters, the *control layer* is the main processing unit of the system that contains the logic of the application, the *eventing layer* creates a simple, topic-based publish/subscribe infrastructure to support eventing, employing *push* technology and the *presentation layer* generates dynamically representations of the connected Smart Meters and their corresponding resources to the Web, enabling uniform interaction with them over a RESTful interface. The gateway is implemented in Java.

We have simulated Smart Meters using Tmote Sky sensor motes running TinyOS. Each mote emulated an actual Smart Meter that can stream the energy consumption of the electrical appliance plugged into it and can switch it on/off. The motes are discovered by the gateway using the mechanism described in Section 2.1. We uploaded on a Web page a WADL file that describes their resources. At runtime, the gateway will aggregate the energy consumption data of the motes, and this data is available as JSON documents, updated continuously.

3.1 Web Mashups

Mashups are Web-based resources that include content and application functionality through reuse and composition of existing resources. The uniform, RESTful interface of our gateway, facilitates the development of smart applications that exploit Smart Meters functionality, from people with very little programming experience, in any language that supports HTTP such as Perl, Php, JavaScript etc. When devices are exposed as Web resources, monitoring rules can be implemented in HTTP, and as an example, we show here a simple rule implemented using a shell script:

```
function check {
   if [ $? -eq 1 ] ; then
       curl -d "interval=1&iterations=1200"
            -X POST localhost:8080/OfficeLaptop/Electricity/Streaming/
   fi
}
curl -s -X GET localhost:8080/OfficeLaptop/State/ $1
check;
```

This rule checks the state of the *OfficeLaptop* and, if it is switched on, then it automatically performs streaming of its electrical consumption every second (*interval=1*), for the next 20 minutes (*iterations=1200*).

To further illustrate the semantics of Web-enabling Smart Meters, we have adapted the Energie Visible[5] project to work with our simulated devices, as shown in Fig. 2. As Energie Visible polls continuously data from a RESTful server, we simply need to point it to our gateway, and it works directly as long

[5] `http://www.webofthings.com/energievisible`

as devices offer their data using the correct JSON syntax. Energie Visible is a mashup developed by Google Web Toolkit that plots sensor data in real-time from energy meters, and illustrates how user interaction with energy data can be simply developed on top of a Web interface.

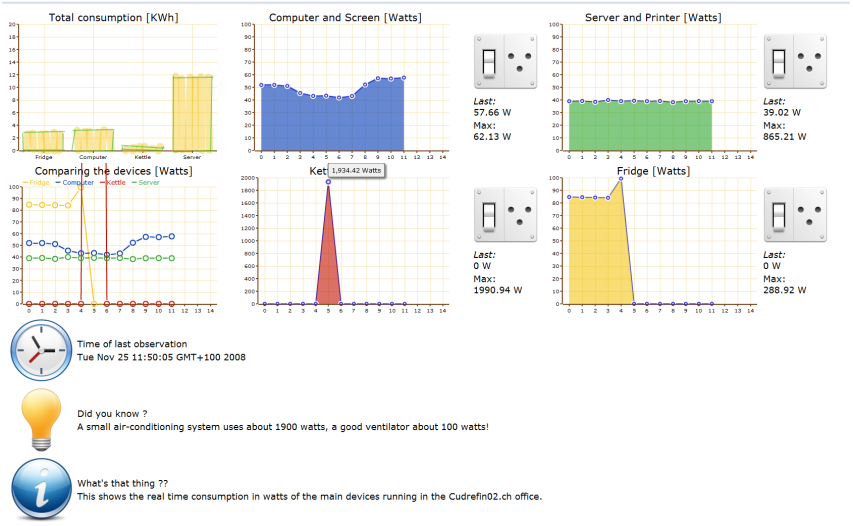

Fig. 2. The user interface of Energie visible. The energy consumption of electrical appliances is shown in real-time.

4 Evaluation

We have performed a preliminary evaluation of our approach. In the first experiment, we measured the time required by the discovery procedure, that is how long it takes for the gateway to discover a varying number of Smart Meters. We performed the experiment six times for each number of sensor motes. The results are shown in Fig. 3 (left). As shown, even with 16 meters operating concurrently, all the devices are discovered within 15 seconds, which is enough especially as this procedure happens only once for each device installed. Besides, in a typical home, there are rarely more than 16 electrical devices operating.

In the second experiment, we tested the gateway in an eventing scenario. Each mote was sending its energy consumption data once per second to the gateway. We then placed a second gateway on the same LAN which subscribed at the first gateway for energy events. Whenever a new energy message was sent by a sensor mote, the second gateway started a timer (t_0, which is the event generation time). The timer was stopped when the second gateway received the notification from the first gateway about the same event (t_1). Therefore the time delay ($t_d = t_1 - t_0$) is the time needed for the first gateway to process and forward events, and is shown in Fig. 3 (right), for a variable number of Smart Meters operating simultaneously.

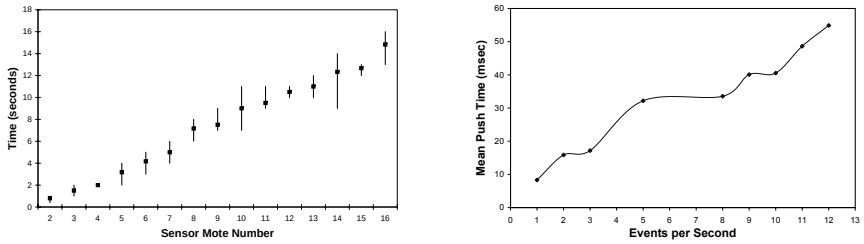

Fig. 3. Smart Meters Discovery Execution Time (Left). Eventing Push Performance (Right).

From the graph, we can see that the gateway performed quite well with a reasonable workload. Even when the gateway received more than 10 messages per second, the event processing and forwarding time was lower than 60 ms to notify subscribers about the energy data. This is largely sufficient when one considers than most smart metering scenarios at a home scale will rarely require gateways to process more than 10 messages per second.

5 Conclusion and Future Work

In this paper, we have shown how the core principles of the modern Web architecture can be exploited to build a scalable infrastructure to support Smart Meters into future houses and buildings. We have illustrated how the uniform and standard Web protocols, can simplify the development of applications that raise awareness about our own energy consumption patterns. We have also shown that the performance of our approach is largely sufficient for scenarios that consider a considerable number of devices. Our future work targets larger scenarios of collecting data from multiple Smart Meters and RESTful gateways, real-time analysis of energy and actual integration of energy data with social networks.

References

1. R. T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, Irvine, California, 2000.
2. D. Guinard and V. Trifa. Towards the web of things: Web mashups for embedded devices. In *Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web, in Proc. of WWW Conference*, Madrid, Spain, 2009.
3. L. Schor, P. Sommer, and R. Wattenhofer. Towards a Zero-Configuration Wireless Sensor Network Architecture for Smart Buildings. In *First ACM Workshop On Embedded Sensing Systems For Energy-Efficiency In Buildings (BuildSys), Berkeley, California, USA*, November 2009.
4. D. Yazar and A. Dunkels. Efficient application integration in ip-based sensor network. In *Proc. of the First ACM Workshop On Embedded Sensing Systems For Energy-Efficiency In Buildings (BuildSys), at SenSys09*, 2009.