

HomeWeb: An Application Framework for Web-based Smart Homes

Andreas Kamilaris*, Vlad Trifa[†] and Andreas Pitsillides*

*Networks Research Laboratory, Department of Computer Science
University of Cyprus, Nicosia, Cyprus

Email: kami@cs.ucy.ac.cy, andreas.pitsillides@ucy.ac.cy

[†] Institute for Pervasive Computing ETH Zurich

MIT SENSEable City Lab

Zurich, Switzerland

Email: trifa@inf.ethz.ch

Abstract—Household appliances are being equipped with embedded micro-controllers and wireless transceivers, offering smart behavior. These augmented appliances form wireless networks and transform residential areas into smart homes. Advancements such as the effective penetration of the Internet in embedded computing and the promising practice of the Web of Things, allow the realization of Web-oriented smart homes. In a previous work, we developed a Web-based application framework for smart homes, supporting concurrent interaction from multiple family members. In this paper, we improve the functionality of our system by including a 6LoWPAN-based wireless sensor network inside the home environment, addressing issues such as device discovery and service description. Web techniques such as HTTP caching and push messaging, facilitate the efficient operation of a fully Web-based smart home. Through a technical evaluation, we show the benefits of directly Web-enabling embedded sensors in terms of performance and energy conservation. The development of a Web-based graphical application abstracts home automation procedure for typical residents.

Index Terms—Web of Things, Smart Home, Wireless Sensor Networks, Web Caching, IPv6, REST.

I. INTRODUCTION

Embedded sensors and wireless sensor networks (WSN) are being deployed around the world in a large scale, measuring with high precision environmental conditions such as temperature and illumination or physical events such as vibration and motion. WSN provide a reliable and extensible solution for real-world deployments.

Household appliances are being equipped with embedded micro-controllers and wireless transceivers, offering smart behavior and communication capabilities. These augmented appliances become able to form wireless networks, transforming residential areas into smart home environments.

High-precision sensors, incorporated in smart homes, provide context-awareness of the environmental conditions that exist at each room of the house. Combining context-awareness with administration of the household appliances that “live” inside the residence, allows advanced home automation. When

electrical appliances are also augmented with individual monitoring and control, efficient energy management is possible.

In the latest years, technologies like sensor networks, RFID, short-range wireless communications and real-time localization are becoming largely common, bringing embedded Internet into commercial use. The *Internet of Things* (IoT) [6] includes technologies and research disciplines that enable the Internet to reach out into the real world of physical objects. The introduction of IPv6 and the efforts for porting the IP stack on embedded devices [4], [9] facilitate the effective penetration of the Internet in embedded computing.

Extending the notion of the IoT, the *Web of Things* (WoT) [19] builds upon the success of the Web 2.0, and reuses well-accepted and understood Web principles to interconnect the expanding ecosystem of embedded devices, built into everyday appliances. It is about taking the Web as we know it and extending it so that anyone can plug devices into it.

In this paper, we address the limitations of our previous work in using Web standards to build the smart home [13]. Our contribution is the use of Web-enabled sensor devices operating in a 6LoWPAN WSN and the improvement of the device discovery procedure considering the IPv6-based wireless network. We compensate the overhead introduced with embedding the IPv6 stack on sensor devices by using HTTP caching to significantly reduce the mean response time for accessing sensor data. Additionally, we investigate how modern Web messaging can be used to incorporate event-based scenarios in the smart home. Finally, we develop a Web-based graphical interface to abstract home automation procedure for typical home residents.

In the following sections we present our work in using solely the Web for interacting with smart devices. In Section II we present related work in our field while Section III provides background information concerning our earlier work in implementing a Web-based smart home. In Section IV we mention the changes we made, to fully transform our smart home into a Web-based smart space. Next, in Section V we describe a Web application we built to facilitate home

automation procedure for residents and in Section VI we perform an evaluation of our system. Finally, in Section VII we discuss the implications of our results and propose issues to be addressed in future work before concluding this paper.

II. RELATED WORK

Science fiction has advertised an idealized vision of a fully integrated smart home, where all the operations of a house can be efficiently controlled by a central application. However, a major impediment in the realization of this scenario has been the proliferation of incompatible standards and protocols used by various device manufacturers, which makes the smooth integration of appliances from different vendors a time- and money-consuming process at best.

To address the heterogeneity of devices, SOAP-based Web services have been proposed as a basis to build an infrastructure for domestic networks [1]. More recently, Priyantha et al. [15] have successfully used Web services to interact with sensor networks. They state that a key challenge in using Web services on embedded sensors is the energy overhead of the structured data formats used in them. Clearly, SOAP/XML are not energy-efficient in such areas. The work in [7] quantifies this statement in terms of time and energy.

Recent approaches suggest REST as the architectural style used in constrained environments. Yazar et al. [20] implement an IP-based sensor network system where nodes communicate their information using RESTful Web services. Similarly, Schor et al. [17] developed a 6LoWPAN-enabled WSN, directly integrated into the IPv6-based future Internet. Sensor nodes, in this scenario, offer RESTful APIs and service discovery is based on Apple's mDNS proposal.

Current and emerging architectures and technologies that are suitable for wireless home area networks (WHAN) have been surveyed in [14]. According to this survey, IPv6 and 6LoWPAN technologies are preferred because of their low cost, ease of installation, flexibility and reliability.

A path towards the integration of things into the Web has been presented in [19], where physical objects are made available through RESTful principles. The concept of physical mashups introduced in [8], builds upon the success of Web 2.0 mashup applications, by proposing a similar approach for integrating real-world devices to the Web. pREST [3] defines a RESTful protocol to bring the simplicity and holistic view of data and services on the Web, to pervasive environments.

In previous work, we combined the principles of the WoT in the area of smart homes [13]. We developed an application framework that enables multiple home residents to interact concurrently with home appliances, and our results indicated a reliable and efficient operation. However, devices were not directly Web-enabled, as a gateway was needed to access them.

A social perspective for a Web-enabled smart home is discussed in [11]. A Web mashup is developed that involves the Web-enabled household appliances of a smart home, in a social networking application. Web 2.0 technologies are utilized through Facebook to transform the interaction with

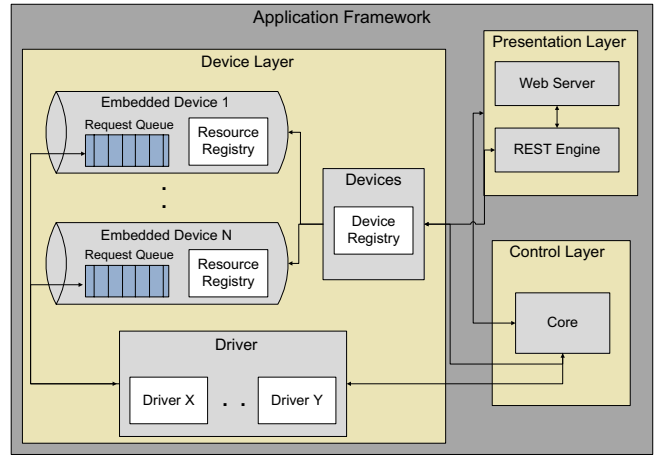


Fig. 1. HomeWeb Architecture.

the home into a shared, social experience. Finally, Web-based smart homes can easily be extended to exploit demand response, which will be a critical functionality in the smart grid of electricity [12].

III. HOMEWEB APPLICATION FRAMEWORK

HomeWeb is a fully Web-based application framework for smart homes [13], where all interactions with embedded devices are done via standard HTTP requests. In this section we describe how to leverage the ubiquitous and well-known Web standards to integrate *by design* household appliances into the Web fabric. Our framework supports concurrent access by multiple residents and facilitates the development of ubiquitous applications by the habitants, who may probably have no programming experience.

REpresentational State Transfer (REST) [5] has been proposed for Web-based interaction with household appliances as it is a lightweight architectural style that defines how to use HTTP as an application protocol. It advocates in providing Web services modeled as *resources*, identified by Unique Resource Identifiers (URI). Resources can only be manipulated by the methods specified in the HTTP standard (e.g. GET, PUT, POST, DELETE), under a uniform interface. REST promotes the practice of Resource Oriented Architectures (ROA) [16], in order to provide and connect together services on the Web. REST guarantees interoperability, loose-coupling and a smooth transition from the Web to home environments.

Web services tend to fall into one of two camps: Big Web services (WS-*) and RESTful Web services. WS-* [2] are a set of complex standards and specifications for enterprise application integration. We believe that RESTful Web services are more appropriate for resource-constrained, ad hoc environments due to their simplicity and flexibility.

Figure 1 depicts the architecture of HomeWeb. It follows a modular architecture and is composed of three principal layers: *Device Layer*, which is responsible for the management and control of embedded devices, *Control Layer*, which is the central processing unit of the system and *Presentation Layer*,

which generates dynamically a representation of the available devices and their corresponding services to the Web, enabling the uniform interaction with them over a RESTful interface.

Each time a new device is discovered, a new thread dedicated to the device is created. In order to provide multi-user support, we attach a *Request Queue* to each thread, to enqueue concurrent requests to it. Requests are stored in a FIFO manner and are transmitted sequentially to the device.

The uniform RESTful interface facilitates home automation through the development of flexible applications, which can easily be created on top of heterogeneous sensor devices. Our Web-oriented framework supports simple creation of Web mashups and advanced rules in any programming language that supports HTTP. Web mashups are extended into physical mashups by exploiting real-world services offered by physical devices and combining them using the same tools and techniques of classic Web mashups [8].

Sensor nodes are used to emulate household appliances, as they can be programmed easily and they offer networking and sensing capabilities. In our previous work, we did not directly embed Web support in our sensor devices, but hid the native programming model of TinyOS¹ behind a RESTful API using a custom driver inside the application framework. Additionally, our device discovery procedure was simplistic and not based on existing standards. These constraints decreased the overall flexibility of our system and hindered our vision for a truly Web-based smart home.

IV. WEB-ENABLING SENSOR DEVICES

The recent progress in embedded IPv6 makes it possible to run IPv6 applications directly on sensor nodes [9]. We believe that Internet technology, utilizing IPv6, will become the future standard in home automation.

An IP-based solution for home automation offers equivalent performance with related traditional home automation standards, while concepts from the Web provide benefits to developers and users [14]. An IPv6-enabled WSN is mature and future-proof, low-cost, with easy installation using wireless embedded devices, auto-configurable, offers wide-scale connectivity and natural user interaction.

We developed an IPv6-enabled WSN, consisting of Telosb sensor motes equipped with temperature, humidity and light sensors. Their sensing capabilities can be accessed using standard HTTP requests thanks to a RESTful interface. Our implementation is based on Blip², which is an implementation of the 6LoWPAN stack for TinyOS 2.x. 6LoWPAN is an adaption layer that allows efficient IPv6 communication over IEEE 802.15.4.

A. Device Discovery

Since we envision Web-enablement of embedded devices, we need to follow Web-based approaches also for discovering these devices. We alter the device discovery procedure in [13]

and we extend it to take into account the IPv6 operation environment.

Our new device discovery protocol is similar to WS-Discovery³, which is a multicast discovery protocol to locate WS-* services on a local network. We adapted this protocol for RESTful Web services by transmitting a single URL instead of a heavy SOAP/XML payload in the multicast message. This URL points to a Web page, where the services offered by the device are described. As soon as a sensor mote joins the 6LoWPAN network, it announces its availability by means of the multicast message. The application framework listens at the pre-specified multicasting channel and it acknowledges the newly found sensor device immediately. When the sensor receives the acknowledgment, it starts operating normally as an embedded Web server.

B. Service Description

As we mentioned in the previous subsection, an embedded device needs to transmit a URL, in which the services offered by it are described. We followed a standardized, Web-based way to perform resource description, in order to achieve high interoperability with heterogeneous devices and services.

We adopted Web Application Description Language (WADL⁴). WADL is an XML-based file format that provides a machine-readable description of HTTP-based Web applications, and can be considered as the RESTful equivalent of Web Services Description Language (WSDL⁵), which is a standard for describing SOAP-based Web services.

WADL is intended for applications that are based on the existing architecture of the Web, and it is meant as a platform- and language-independent way of describing services, to promote reuse of applications. We used WADL to model the resources (services) provided by an embedded device and the relationships between them.

C. Web Messaging for Event-driven Scenarios

Since sensor motes operate as Web servers, interaction with them follows the classical client-server model employing *pull* technology, in which the request for the transmission of information is initiated by the client, which is the application framework in our case. This model is appropriate for ad hoc interactions with sensors and actuators and for environmental monitoring which happens on demand.

However, the client-server model is not efficient in event-based scenarios. These scenarios include events that are sensed sporadically when something important happens e.g. detection of fire or the opening of a door.

Push technology describes a style of Internet-based communication where the service request is initiated by the publisher, which is the sensor device in our case. We build upon recent developments in Web push techniques and extend them for

¹<http://www.tinyos.net/>

²[http://docs.tinyos.net/index.php/BLIP Tutorial](http://docs.tinyos.net/index.php/BLIP_Tutorial)

³<http://docs.oasis-open.org/ws-dd/discovery/1.1/os/wsdd-discovery-1.1-spec-os.html>

⁴<https://wadl.dev.java.net/>

⁵<http://www.w3.org/TR/wsdl>

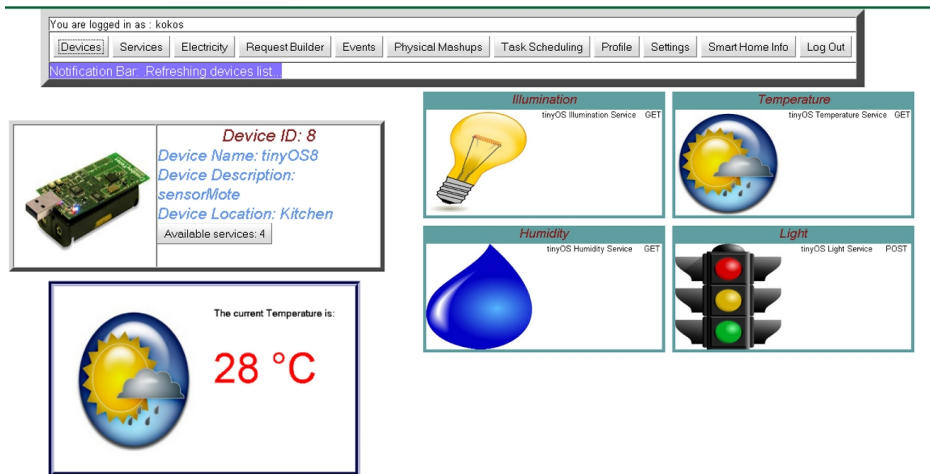


Fig. 2. Snapshot of the Web-based Graphical Interface.

embedded devices with a RESTful messaging system, in order to efficiently integrate event-driven services in our system.

In a previous work, we developed the RESTful Message System (RMS) [18], which is a lightweight publish/subscribe messaging suited for sensor devices. We apply RMS to our IPv6-enabled sensors by enhancing them with a simple RESTful API to use the eventing system.

In case the application framework wants to receive notifications about an event from a sensor mote, it creates a new subscription by POSTing an HTTP request to this particular sensor. Whenever a new event is sensed, it will be POSTed back to this subscriber at a callback URL, specified by the subscriber at the subscription message.

D. HTTP Caching

Adding IPv6 capabilities on sensor motes has apparent effects on their performance and energy consumption. However, we can decrease these disadvantages by exploiting the HTTP caching feature.

HTTP has a very thorough and well supported caching mechanism. A Web cache sits between the (origin) Web servers and clients, and monitors incoming requests, saving copies of the responses for itself. Then, if there is another request for the same URL, it can use the cached response, instead of asking the origin server for it again. Web caches reduce latency because the request is satisfied directly from the cache, and also reduce network traffic as the amount of bandwidth used by a client is reduced. Gateway caches are typically deployed by Web masters, to make their sites more scalable, reliable and better performing.

For our purposes, origin Web servers are the sensor devices, clients are the residents of the smart home and our application framework includes a gateway cache, to accelerate performance and save sensors' energy. Our caching mechanism is used only for GET requests and it uses the expiration model, which is a way for the server to say how long a requested

resource stays fresh. We will test the performance of the caching mechanism in Section VI-A.

V. A WEB-BASED GUI FOR THE SMART HOME

Since home residents are not expected to have programming knowledge, we developed a Web-based graphical interface to facilitate home automation and help residents to visualize their environmental context. This client application was implemented in JavaScript, using the Google Web Toolkit (GWT⁶). It communicates with our application framework through its RESTful interface, exposed by the presentation layer.

The client application is split into a number of graphical interfaces. *Devices* interface lists the available sensor devices, *Services* presents the services offered by some specific device while *Request Builder* is an easy way to create Web-based requests for some device just by filling a form.

Events interface enables notifications to be triggered when some event happens. A notification can be performed through SMS, email or through a Web hook⁷. For example, the resident can program the sending of an email as soon as the temperature falls below 20 degrees Celsius.

Physical Mashups interface allows the easy creation of mashups in a few clicks e.g. to light the LEDs of a sensor mote, when the humidity in the smart home exceeds 60%. Furthermore, *Task Scheduling* allows the automatic execution of some service in a future time.

Profile interface manages all the previously created event notifications and mashups, allowing residents to view, edit or delete their smart rules. Finally, *Settings* holds all the general configurations of the application, including an option to automatically query sensor devices in specified intervals.

Figure 2 shows a snapshot of the application, in which the services offered by a particular device are presented. The user can view the current state of each service just by hovering over it with the mouse pointer.

⁶<http://code.google.com/intl/webtoolkit/>

⁷www.webhooks.org/

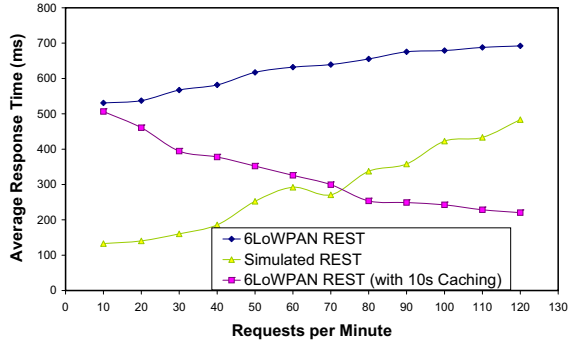


Fig. 3. Comparison of average Request/Response times.

VI. TECHNICAL EVALUATION

In this section, we perform a technical evaluation of our application framework, when combined with IPv6-enabled sensor nodes. We consider three different scenarios to test our system. At the first, we assume the existence of a large family, which interacts concurrently with the framework. In the second, we examine the multihop performance of our 6LoWPAN network in a large home and finally, in the third scenario, we investigate the event-based behavior of our system.

A. A Family with many Members

In this experiment we created a simulation, in which multiple family members are interacting with their home through the Web, by sending concurrent requests to the application framework. The home residents are randomly selecting sensor nodes and RESTful Web services, offered by these devices.

We performed different tests with variable numbers of family members. We employed four Web-based, IPv6-enabled sensor nodes at each test, deployed in a star topology with 30 cm distance from the base station. Each test ran for five minutes. We measured the amount of time needed, from the creation of a request from a family member to the arrival of the response, transmitted by the sensor device. Request queues were set for 600 ms, which was considered the most appropriate value to avoid overloading sensors with requests.

Figure 3 shows a comparison of the results obtained from this experiment and the results obtained in [13] for the same simulation in which simulated REST was used, without the IPv6 overhead. Simulated REST is of course much more efficient, especially in light workload. In this case, simulated REST is almost four times faster. In heavy workload (e.g. 120 requests per minute), the difference in performance is not so clear, being less than 1.5 times. This happens because the blip 6LoWPAN implementation is more reliable than our earlier approach and very few failed attempts are encountered. The difference in transmission failures is shown in Figure 4. While failures reach 2% in simulated REST, they remain below 0.7% in the 6LoWPAN case.

Even though simulated REST outperforms 6LoWPAN REST in normal conditions, this is not the case when we exploit HTTP caching. Carefully examining Figure 3, we can

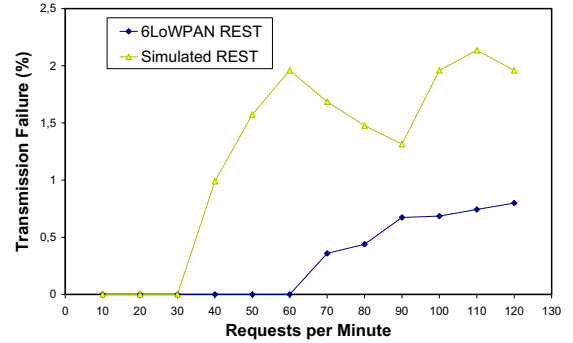


Fig. 4. Transmission Failures.

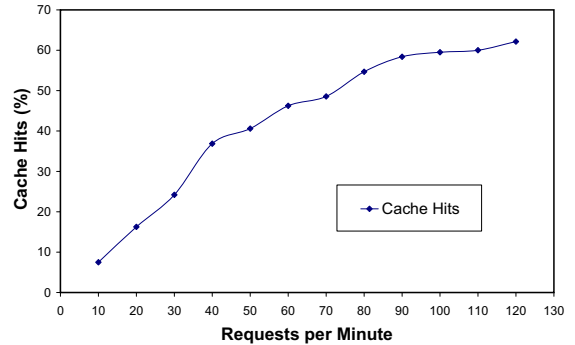


Fig. 5. Cache Success based on number of requests per minute.

observe the results when the framework’s gateway cache is set for 10 seconds freshness time. In light workload scenarios, cache does not influence request/response time significantly as cache hits are minimum. However, as the workload becomes heavier, cache hits increase and the average request/response time is reduced. When there exist more than 70 requests per minute, 6LoWPAN REST with 10 seconds caching needs less time than simulated REST. This time difference becomes 2.2 times bigger in the 120 requests-per-minute test. Efficiency in this case depends on the percentage of cache hits in each test. The correlation between cache hits and requests per minute is depicted in Figure 5. Obviously, cache hits and consequently energy conservation and performance would be increased if we use larger freshness time. This depends on the specific requirements of each application.

B. A large Smart Home

In our earlier work and in the previous experiment, we considered a WSN in a star topology. However, this topology does not cover a typical smart home as the indoor range of a typical sensor node is between 20 and 30 meters. For large homes we must consider multihop topologies.

We deployed our Web-based IPv6-enabled WSN in an experimental smart home of around 100 m². We tested the execution time of the RESTful services that are supported by our Telosb sensor nodes in a multihop scenario.

The results can be observed in Figure 6, for distances up to three hops. We discovered that there was not any significant

time variation between the different services. The completion times in our approach are considered satisfactory, being less than a second, even in the case of three-hop routing.

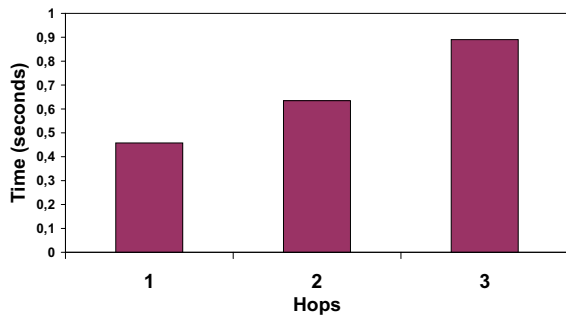


Fig. 6. RESTful 6LoWPAN Network Multihop Completion Time.

C. Detecting Motion in the Smart Home

To test the Web messaging technique of our system in event-based scenarios using push technology, we interfaced our Telosb sensor motes with Zilog's ZMOTION⁸ detection modules. These modules are fully functional motion detection solutions ideal for occupancy and proximity detection applications. They provide an output signal when motion is detected.

We configured our application framework to POST a subscription HTTP request to all sensor motes which included motion detection and consequently supported RMS. Whenever motion was detected, it was POSTed back to the framework (subscriber) at a specified callback URL.

We performed a number of trials in one-hop topology, to identify that just 300-600 ms were needed, from the time motion from humans was performed, until the event notification reached the application framework. Through RMS push technique, advanced performance and energy conservation is achieved, as we avoid polling the sensor devices with continuous request messages.

VII. CONCLUSION

In this paper, we have extended our Web-oriented application framework into a fully-functional, Web-based smart home. We exploited recent advancements in IPv6 and 6LoWPAN technologies to enhance the functionality of our framework with directly Web-enabled sensor devices. Our technical evaluation indicates that the application of Web technologies such as HTTP caching and push techniques in the WSN domain can improve performance while wide-scale connectivity and interoperability are guaranteed.

As future work, we plan to incorporate more advanced technology in the HomeWeb, such as residential smart meters, lighting, HVAC control etc., enabling all these technologies as first-class citizens of the Web. Our efforts will be dedicated towards energy-efficient, sustainable, flexible and secure smart homes that would operate in a true Web environment.

⁸www.zilog.com/docs/PB0223.pdf

Smart homes have an important role to play in future urban environments. The increasing urbanism implies new challenges for the future citizens. The WoT constitutes a promising practice towards the vision of a real-time digital city [10]. Web-based smart homes can be considered as a real-time platform, for engaging people to sense and shape their cities.

REFERENCES

- [1] M. Aiello. The role of web services at home. In *IEEE Web Service-based Systems and Applications (WEBSA)*, page 164, 2006.
- [2] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services: Concepts, Architectures*. Springer, 2004.
- [3] W. Drytkiewicz, I. Radusch, S. Arbanowski, and R. Popescu-Zeletin. pREST: a REST-based protocol for pervasive systems. In *IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, pages 340–348, October 2004.
- [4] A. Dunkels, T. Voigt, and J. Alonso. Making TCP/IP Viable for Wireless Sensor Networks. In *Proc. EWSN 2004*, Berlin, Germany, 2004.
- [5] R. T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, California, 2000.
- [6] N. Gershenfeld, R. Krikorian, and D. Cohen. The Internet of Things. *Scientific American*, 291(4):76–81, October 2004.
- [7] C. Groba and S. Clarke. Web services on embedded systems - a performance study. *Pervasive Computing and Communications Workshops (PERCOM Workshops)*, 2010 8th IEEE International Conference on, pages 726–731, March 2010.
- [8] 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.
- [9] J. W. Hui and D. E. Culler. IP is dead, long live IP for wireless sensor networks. In *SensSys '08: Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 15–28, New York, USA, 2008.
- [10] A. Kamilaris, N. Iannarilli, V. Trifa, and A. Pitsillides. Bridging the Mobile Web and the Web of Things in Urban Environments. In *Urban Internet of Things Workshop*, at *IOT 2010*, Tokyo, Japan, Nov. 2010.
- [11] A. Kamilaris and A. Pitsillides. Social Networking of the Smart Home. In *21st Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2010)*, September 2010.
- [12] A. Kamilaris and A. Pitsillides. Exploiting Demand Response in Web-based Energy-aware Smart Homes. In *First International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies (Energy 2011)*, Venice, Italy, May 2011.
- [13] A. Kamilaris, V. Trifa, and A. Pitsillides. The Smart Home meets the Web of Things. *International Journal of Ad Hoc and Ubiquitous Computing (IJAHUC)*, Special issue on *The Smart Digital Home [To Appear]*, 2011.
- [14] M. Kovatsch, M. Weiss, and D. Guinard. Embedding Internet Technology for Home Automation. In *Proceedings of ETFA 2010 (IEEE Conference on Emerging Technologies and Factory Automation)*, Bilbao, Spain, September 2010.
- [15] N. B. Priyantha, A. Kansal, M. Goraczko, and F. Zhao. Tiny web services: design and implementation of interoperable and evolvable sensor networks. In *Proceedings of the 6th ACM conference on Embedded network sensor systems (SenSys)*, pages 253–266, New York, USA, 2008. ACM.
- [16] L. Richardson and S. Ruby. *RESTful Web Services*. O'Reilly, 2007.
- [17] 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, 2009.
- [18] V. Trifa, D. Guinard, V. Davidovski, A. Kamilaris, and I. Delchev. Web Messaging for Open and Scalable Distributed Sensing Applications. In *International Conference on Web Engineering (ICWE 2010)*, pages 129–143, Austria, June 2010.
- [19] E. Wilde. Putting things to REST. Technical Report UCB iSchool Report 2007-015, School of Information, UC Berkeley, November 2007.
- [20] D. Yazar and A. Dunkels. Efficient Application Integration in IP-based Sensor Networks. In *First ACM Workshop On Embedded Sensing Systems For Energy-Efficiency In Buildings (BuildSys)*, Berkeley, California, 2009.