ENABLING SMART HOMES USING WEB TECHNOLOGIES

Andreas Kamilaris

A Dissertation

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

at the

University of Cyprus

Recommended for Acceptance

by the Department of Computer Science

December, 2012

APPROVAL PAGE

Doctor of Philosophy Dissertation

ENABLING SMART HOMES USING WEB TECHNOLOGIES

Presented by

Andreas Kamilaris

Research Supervisor

ΠΡΟΣΩΠΙΚΑ ΔΕΔΟΜΕΝΑ

Andreas Pitsillides

Committee Member

ΠΡΟΣΩΠΙΚΑ ΔΕΔΟΜΕΝΑ

Vasos Vassiliou

Committee Member

ΠΡΟΣΩΠΙΚΑ ΔΕΔΟΜΕΝΑ

George Pallis

Committee Member

ΠΡΟΣΩΠΙΚΑ ΔΕΔΟΜΕΝΑ

Giuseppe Anastasi

Committee Member

ΠΡΟΣΩΠΙΚΑ ΔΕΔΟΜΕΝΑ

Ioannis Stavrakakis

University of Cyprus

December, 2012

ENABLING SMART HOMES USING WEB TECHNOLOGIES

Andreas Kamilaris

University of Cyprus, 2012

Sensors and wireless sensor networks are being deployed around the world, measuring local and global environmental conditions. Their advanced sensing functionalities enable context-aware ubiquitous platforms, middleware and applications to proliferate.

Residences are transformed into smart homes, incorporating embedded sensors and pervasive technology. Moreover, residential smart meters, smart power outlets and smart appliances have appeared in the market, facilitating electricity metering and control of individual electrical appliances, extending homes into smart energy-aware environments. Smart metering provides energy awareness to home residents, and new generations of energy recording devices transform the energy conservation initiatives inside the smart home into a simple task.

In an idealized vision of a fully integrated smart home, all the operations of a house can be efficiently controlled by a unified smart ubiquitous application. However, we are far from realizing this scenario. A main barrier is the proliferation of incompatible standards and protocols used by various device manufacturers, which makes the smooth integration of appliances from different vendors a complex process. Having a heterogeneous ecosystem of devices, implies that even the development of simple applications requires advanced programming skills and considerable time.

In recent years, technologies like short-range wireless communications, RFID and real-time localization are becoming largely common, allowing the Internet to penetrate into the real world of physical objects. The Internet of Things allows household devices that live inside smart homes

to seamlessly communicate through the Internet while the forthcoming Web of Things ensures interoperability at the application level through standardized Web technologies and protocols.

In this thesis, we exploit these new technological possibilities to bring smart homes towards the Web, achieving high interoperability and flexibility. By employing standardized, well-known, reliable and scalable Web techniques, hardware and software heterogeneity between the embedded devices of the smart home can be addressed.

We present the development of a Web-based application framework for smart homes, supporting concurrent interaction from multiple family members. By employing intermediate request queues, associated with the physical devices of the smart home, our analysis shows that we can mask transmission failures and faults that occur in the wireless environment, thus enhancing the performance of smart home operations by means of fast retransmissions, load balancing and request priority techniques. In our analysis, we also derive formulas for estimating the response time of requests and for setting the request queue retransmission interval, an important design parameter of the system. In this way, reliability and timely responses from the devices are ensured.

We demonstrate that, by using the Web as application layer, flexible applications for smart homes can be built, on top of heterogeneous embedded devices, with little effort. In this context, Web mashups may be 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. Urban mashups may also be developed, for adapting to the high dynamics and unpredictability of a future urban environment.

We address many issues related to Web-enabling household devices, from their local discovery and service description to the uniform interaction with them. Our technical evaluation indicates that the process of Web-enabling physical devices offers satisfactory performance, mainly in terms

of response time and energy consumption, while modern Web techniques such as Web caching and event-based Web messaging can contribute in facilitating smart home operations.

The initiative of enabling smart homes to the Web unfolds novel applications. By means of these pervasive applications, it is demonstrated that Web-based smart homes have the potential to provide flexible solutions to challenges such as home automation, energy awareness and energy conservation. At first, we show how we can extend our framework into an energy-aware Web application. We explain how we can easily define energy-efficient smart rules, by following the physical mashup paradigm.

Afterwards, we demonstrate the feasibility of merging Web-based smart homes with online social networking platforms. This merging can help social groups such as families, workers and friends, to interact with their common physical environment through their favorite social networking application. Moreover, to motivate people to become more energy-aware and reduce their electrical consumption, we created a social competition between neighboring flats in large residential blocks. Our findings show that people react positively in such energy-saving initiatives. In a similar context, we introduce Social Electricity, a Facebook application allowing people to compare their domestic electricity footprint with their friends and their neighbors in a wide scale. Social Electricity has been awarded the first prize in a prestigious international competition about green ICT applications. In general, social comparisons have the potential to increase the environmental awareness of citizens, helping them to understand their consumption and save energy.

Then, we illustrate how Web-based smart homes can be seamlessly integrated to the forthcoming smart grid of electricity. We propose an architecture for enabling the interaction between smart homes and smart grid controllers, through the Web. This architecture is used in two case

studies we performed. In the first, demand response programs from electric utilities can be harnessed through the Web, to schedule electricity-related tasks in low-tariff hours of the day. In the second, a load shedding scenario is considered for maintaining frequency stability by removing intentionally small loads from energy-aware smart homes that participate in the smart grid.

Finally, we consider some issues beyond the smart home environment, which involve the generalization of the application framework for other indoor and outdoor pervasive scenarios, the global, real-time discovery of environmental services through the Web and also the advanced inference of environmental knowledge in urban environments, following the urban mashup paradigm. We briefly touch upon security both inside the smart home environment and between the home and third parties, defining some basic guidelines that need to be followed.

Our work constitutes a research towards a flexible, interoperable, energy-efficient and sustainable digital future. Smart homes have the potential to effectively contribute in this effort, using the Web as a platform.

DEDICATIONS

To my father, for motivating and encouraging my research from the heavens.

To my family, for being always next to me, during the happy and bad moments of this work.

To my girlfriend, who showed great patience in my overloaded daily schedule.

To my friends and relatives, for their psychological encouragement and support.

ACKNOWLEDGEMENTS

This dissertation would not have been possible without the guidance, the advice and the support of several individuals who contributed in the completion of this study. First and foremost, I would like to express my gratitude to Dr. Andreas Pitsillides, Professor at the Department of Computer Science, University of Cyprus whose help and encouragement I will always remember and appreciate. Heartfelt thanks to him, for not only supporting me a supervisor, but also as a friend. Moreover, I would like to express my special appreciation to Dr. Pavlos Antoniou, who encouraged me at the beginning of my research. Moreover, this work would not have been realized without a fruitful collaboration with various students of the Department of Computer Science, University of Cyprus namely Michalis Yiallouros, George Taliadoros, Diomidis Papadiomidous, Giannis Kitromilides, Marios Michael, Andreas Loizou and Koula Papakonstantinou. I would like to thank them all here. Furthermore, I would like to express my special thanks to researcher Yiannis Tofis, PhD student at the KIOS Center for Intelligent Systems and Networks, for his nice collaboration towards integrating energy-aware smart homes with the smart grid through the Web.

I am also grateful to Dr. Vlad Trifa, co-founder and chief product officer of EVRYTHNG company, for giving me the initial stimuli and motivation to explore this challenging and interesting research area of the Web of Things, and for providing lots of good ideas. His expertise in the field of pervasive computing and Web technologies was quite valuable to develop this thesis.

I would like to thank Dr. Vasos Vassiliou and Dr. George Pallis, both members of the faculty of the Computer Science Department, University of Cyprus for serving as readers of my thesis and for providing valuable comments. Special thanks also to the external committee members of

this dissertation Dr. Giuseppe Anastasi and Dr. Ioannis Stavrakakis, who honored this work by visiting Cyprus, giving valuable feedback in order to improve it.

Finally, I would like to acknowledge the academic and technical support of the Department of Computer Science, University of Cyprus and especially the Networks Research Lab (NetRL), which have provided the support and equipment I needed to produce and complete my thesis. I am indebted to all student colleagues, members of NetRL, for providing a stimulating and fun environment in which to grow and learn. A special mention to Christiana Ioannou and Josephine Antoniou for being the best colleagues in my office, as well as to Andreas Panayides, Zinon Zinonos, Marios Koutroullos and Panayiota Nikolaou for the funny moments we shared together.

"This research is based on real evidence."

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

6LoWPAN	IPv6 over Low power Wireless Personal Area Networks
ACK	Acknowledgement Message
API	Application Programming Interface
Blip	Berkeley low-power IP stack
DNS	Domain Name System
DR	Demand Response
EAC	Electricity Authority of Cyprus
FIFO	First In, First Out
GPS	Global Positioning System
GUI	Graphical User Interface
GWT	Google Web Toolkit
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
ICT	Information and Communication Technology
IoT	Internet of Things
IP	Internet Protocol
IPv6	Internet Protocol version 6
JSON	JavaScript Object Notation
MIME	Multipurpose Internet Mail Extensions
M2M	Machine-to-Machine
PHEV	Plug-in Hybrid Electric Vehicle

REST	REpresentational State Transfer
RFID	Radio-Frequency IDentification
RMS	RESTful Message System
ROA	Resource-Oriented Architectures
SCADA	Supervisory Control And Data Acquisition
SLA	Service-Level Agreement
SNS	Social Networking Sites
TCP	Transport Control Protocol
TLS	Transport Layer Security
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WADL	Web Application Description Language
WoT	Web of Things
WSN	Wireless Sensor Network
WS-*	Big Web services
XML	Extensible Markup Language

LIST OF SYMBOLS

α	Request Queue Retransmission Interval (in ms)
λ	Arrival Rate of Requests at the Request Queue (in requests per second)
λ_c	Critical Arrival Rate for Stability (in requests per second)
c	Specifies the sensitivity of the retransmission interval α
n	Maximum Retransmission Attempts (in no. of retransmissions)
p_f	Probability of Transmission Failures
q	Waiting Time for some Request at the Request Queue (in ms)
R	Residual Service Time (in ms)
RTT	Round-Trip Time (in ms)
S_q	Current Size of the Request Queue (in no. of requests)
Simulated REST	A Simulated Version of REST without IPv6 on Telosb Sensor Motes
6LoWPAN REST	A Version of REST with IPv6 on Telosb Sensor Motes

Chapter 1

Introduction

Embedded physical devices, such as household appliances are becoming smarter and smarter. They are equipped with embedded microprocessors and wireless transceivers, offering limited communication capabilities and providing smart behavior. Everyday objects are fitted with small, cheap mobile processors, sensors and actuators.

Sensors and wireless sensor networks are being deployed in smart home solutions, measuring with precision the environmental conditions inside the home environment. Their advanced sensing functionalities and their increasing accuracy enable the development of smart home applications that offer advanced automation. Residences are transformed into smart homes, incorporating embedded sensors and actuators, and pervasive technology.

This merging of computing with physical objects introduces the concept of information appliances [20], defined as devices or machines, designed to perform some specific functionality but are usable, at the same time, for the purposes of computing. Typical examples of computing devices inclue smart phones, embedded sensors and actuators, radio-frequency identification (RFID) chips and smart cards. As Donald A. Norman [122] points out, *"the trend in computing is towards*

simplicity through specialization". This trend seems to justify Mark Weiser's vision of the *Disappearing Computer* [175], according to which *"the most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it"*. These augmented appliances, when interconnected, they can form wireless networks, extending residential areas into smart pervasive environments.

In the near future, homes will offer new automation possibilities to their residents, increasing their comfort level. Changes will happen to the way people live and interact with their home environments, when technology recedes into the background of their lives, when information processing is thoroughly integrated into everyday objects and activities. For example, coffee machines appeared that are able to prepare coffee automatically, according to user preferences. Fridges that offer dedicated programming interfaces for their control are being produced.

By now, residential smart meters have been introduced in our lives as sensor devices that measure in small time intervals the energy consumption of a house. Moreover, smart power outlets are devices that measure the consumption of individual electrical appliances and control their operation in real-time. It is expected that soon, smart appliances would handle automatically their intended operation and function optimally in order to save energy and perform their task effectively. They may even take advantage of the functionality of the forth-coming smart grid of electricity to synchronize their operation with its current state. For example, they may respond to pricing signals and decide when it is most economical to operate. In general, the practice of equipping smart homes with smart meters, smart appliances and smart power outlets, enables the extension of smart homes into energy-aware environments. Towards this end, this thesis will analyze the environment described above and investigate and offer effective solutions.

The rest of the chapter is organized as follows: Section 1.1 states the problems encountered in this thesis and the motivation for addressing them while Section 1.2 explains the general approach

for tackling those issues. Then, Section 1.3 discusses the reasoning for adopting some particular approaches and technologies. Section 1.4 identifies the unique contribution of the thesis while Section 1.5 acknowledges the scientific contribution of other researchers to this thesis. Finally, Section 1.6 lists publications of the author relevant to the thesis and Section 1.7 discusses the general organization of the thesis.

1.1 Problem Statement and Motivation

In an idealized vision of a fully integrated smart home, all the operations of the house can be efficiently controlled by a smart ubiquitous application. However, we are far from realizing this scenario. A main barrier is the proliferation of incompatible standards and protocols used by various device manufacturers, which makes the smooth integration of appliances from different vendors a very complex process. Some well-known solutions for home automation are X10, KNX, ZigBee, digitalSTROM, Z-Wave, INSTEON, Wavenis, CEBus and LonWorks. Each of these technologies has an important portion of the global home automation market. Having a heterogeneous ecosystem of devices, implies that even the development of simple applications requires advanced programming skills and a considerable amount of time.

Embedded technology, based on multi-hop wireless communications, is being integrated lately in smart environments such as smart homes. However, even though this technology is becoming mature and highly reliable, it still faces issues such as device failures, transmission losses and unexpected delays. Home residents and family members have advanced needs, demanding direct access to their home devices easily and fast. These people may need to interact simultaneously with their smart home environment. Moreover, smart home applications that target home automation and energy conservation cannot tolerate faults and errors. Hence, application frameworks for

smart homes need to compensate problems at the physical, ad hoc, resource-constrained environment and also to support multiple users who may interact concurrently with their smart home.

Reliability and time performance are crucial in multi-user smart home applications that deal with the safety of the house and its tenants or health scenarios. For example, faulty electrical appliances may need to be turned off immediately, to avoid the possibility of fire or the alarm must be triggered when a thief breaks into the house. In this case, support for prioritized requests and load balancing are important mechanisms that could address such issues, as we will be analyzing later in the thesis.

Furthermore, application development from third parties and end users for automating home environments is currently a complicated procedure, since the user needs to study and understand the vendor-specific proprietary protocols and technologies employed in each smart home solution. For example, X10, KNX, Z-Wave and ZigBee define specific wireline or wireless communications protocols for interacting with physical devices that operate with these standards.

Open, standardized approaches or flexible techniques for encouraging and facilitating the creation of smart ubiquitous applications, by home residents who can have very little or even no programming experience, do not normally exist. The access barrier that allows people to develop their own composite applications must be lowered.

Technological advancements such as sensor networks, short-range wireless communications and RFID allow the Internet to penetrate in embedded computing. The *Internet of Things* (IoT) [62, 115] was proposed a decade ago for connecting physical devices at the network level. Recent efforts for porting the Internet protocol (IP) stack on embedded devices [49, 86] and the introduction of IPv6, which provides extremely large addressing capabilities, are expected to facilitate the merging of the physical and the digital world through the Internet, where everyday objects are uniquely addressable and interconnected.

Porting the TCP/IP stack on embedded devices caused the initial concerns of the academic community, whether this would be feasible in terms of performance. These concerns centred on critical scenarios with heavy workloads and minimal response times, as even milliseconds of delay or message transmission failures are unacceptable. Furthermore, some concerns involved whether the IP overhead could affect the energy behavior of home devices. Of course, residents would not be willing to change frequently the batteries of their wireless sensors and actuators. We shall show later in this thesis that the IoT has only a small impact on the performance of physical devices, which is compensated by the benefits acquired for reasons of flexibility and interoperability. Besides, as we would also show, this impact becomes negligible when we employ some basic techniques of the Web of Things.

Building upon the notion of the IoT, the *Web of Things* (WoT) was recently proposed [178, 76], promising to seamleslly interconnect devices and their functionalities. The WoT reuses well-accepted and understood Web principles to interconnect the expanding ecosystem of embedded devices. While the IoT focuses on interconnecting heterogeneous devices at the network layer, the WoT can be seen as a promising practice to achieve interoperability at the application layer. It is about taking the Web as we know it and extending it so that anyone can plug devices into it.

The WoT follows the concept of resource-oriented architectures and REpresentational State Transfer (REST) [59, 142], proposing the use of uniform patterns, influenced by the HTTP protocol, for managing and communicating with pervasive smart home devices. REST is an architectural style that defines how to properly use HTTP as an application protocol.

While this could provide enhanced interoperability between heterogeneous devices and services, a common belief is that Web standards are inappropriate for building pervasive applications. In addition, many issues arise for enabling the devices in a dynamic, plug and play way to the Web. Issues such as local device discovery, service description, uniform and transparent interaction with

embedded devices, sharing of their services, notifications in event-based scenarios etc. need to be addressed effectively and efficiently.

Finally, Web-enabled smart homes have a crucial role to play in current and future technological and societal challenges. Energy awareness of home residents, energy conservation through advanced automation and social approaches, the forth-coming integration of smart homes to the smart grid of electricity, even the contribution of smart homes towards the digitization of the urban environment are some of the many challenges which are encountered by smart homes and their residents. This important role of smart homes in relation to the physical and urban environment needs to be investigated, highlighted and promoted. Technologies, protocols and architectures need to be defined, which permit the smooth and flexible adaptation of smart home solutions to these challenges, ensuring satisfactory levels of security and user privacy. Numerous case studies need to be developed to illustrate the above and hence encourage the adoption of Web technologies in smart homes.

Given the above, the main research challenges encountered in this PhD thesis, for which solutions will be sought, are:

- **Heterogeneity of hardware and software** of smart home solutions and technologies, as well as incompatible, vendor-specific approaches for home automation.
- **Support of multiple family members** who may interact concurrently with their smart home environment.
- **Failures** concerning device unavailability and message transmission losses, which may happen in the unpredictable embedded home environment and decrease the reliability of the smart home operations.

- **Possible degradation of performance**, caused by utilizing the IoT for interconnecting physical devices and their services, mainly due to the IP overhead of exchanged messages.
- **Performance in terms of response time**, especially in scenarios involving house safety and health applications.
- **Energy consumption performance** of embedded home devices, in particular when they incorporate the IP stack.
- **Complicated, vendor-specific closed smart application development**, not based on any standards or well-known techniques.
- Issues encountered during the process of dynamically **enabling physical devices and their services to the Web**.
- **The role of smart homes in current and future technological and societal challenges** and the flexible adaptation of smart homes to these challenges.

1.2 General Approach

In this thesis, in order to address heterogeneity of home embedded devices and smart home technologies, we examine whether the Web can constitute an effective and efficient application layer in future smart homes. Since the Web is ubiquitous and it scales particularly well, we believe it can be harnessed for achieving interoperability in the smart home environment. Furthermore, according to related work [105, 65, 16], an IP-based approach for home automation can offer equivalent performance with related traditional home automation standards, while concepts from the Web bring convenience to developers and users.

Our experimental results and prototypes presented through this thesis, suggest that the WoT is not only a suitable, but actually a desirable medium to build powerful and reliable domestic applications, which match the requirements of future smart homes. REST may be suitable for smart home environments due to its simplicity and flexibility [73]. Embedded devices inside the smart home may expose their functionalities as RESTful Web services, facilitating the interaction with them through standardized HTTP calls.

We exploit the fact that wireless sensor networks (WSN) can provide nowadays a reliable and extensible solution for real-world deployments in smart homes and we take advantage of their wireless, multi-hop networking capabilities to investigate the feasibility of developing smart home applications, which are based on wireless technology. We focus on wireless communications inside the home environment as they constitute a flexible, plug and play approach to accommodate ad hoc networks of sensors, actuators, smart meters and smart power outlets. Their deployment does not require any extensive modifications in existing buildings.

The recent progress in embedded computing makes it possible to run IPv6 applications directly on sensor nodes [49, 86]. Wireless sensor networking based on Internet protocols, is promoted due to the development of IPv6 over Low power Wireless Personal Area Networks (6LoWPAN). 6LoWPAN [107] is an adaption layer that allows efficient IPv6 communication over the IEEE 802.15.4[1] standard. 6LoWPAN proposes a new integration level towards the Web, and ensures interoperability among IP-based embedded devices [16].

The development of a 6LoWPAN-enabled WSN, directly integrated into the IPv6-based future Internet is a feasible task [147, 182]. Therefore, sensors and actuators inside the smart home environment may be transformed into embedded Web servers, satisfying the requests of the residents through the HTTP protocol. 6LoWPAN offers numerous benefits to wireless sensor installations

[1] http://www.ieee802.org/15/pub/TG4.html

including the easy interconnection with other IP networks, the reuse of the existing Internet infrastructure, the application of well-known Internet technologies in the ubiquitous environments and the use of existing power monitoring and diagnostic tools.

Comparing IPv6/6LoWPAN with various existing technologies for smart homes (X10, KNX, ZigBee and digitalSTROM), an IPv6-based WSN, being part of the ever-expanding IP ecosystem, is mature and future-proof, with low cost, easy installation using wireless sensors and actuators, auto-configurable, offering wide-scale connectivity, natural user interaction and medium security [105]. Furthermore, a comparison of IPv6/6LoWPAN with ZigBee, Z-Wave, INSTEON and Wavenis, in terms of physical characteristics, communication modes, networking and security concludes that IP-based solutions offer enhanced quality, security, and interoperability [65].

Recently, Wi-Fi, as an enabling technology for IP-connectivity of embedded devices was explored in [157]. According to the authors, Wi-Fi technology is feasible for the IoT considering the power consumption of devices, the impact of interference and the range performance. Even though Wi-Fi supports only a star topology, an experimental deployment in the paper shows that a single access point can cover a typical home environment adequately. According to the authors, Wi-Fi may be considered as a candidate option for enabling the IoT. As future work, we could compare the performance of Wi-Fi and 6LoWPAN in home environments.

Furthermore, online social networking has become a fundamental part of people's everyday lives, transforming the Web 2.0 into a social Web, in which human social capabilities are boosted. The next big challenge for the Web, after its massive utilization by humans, is to interconnect the physical world. Physical devices, enhanced with embedded microcontrollers, could be an essential part of the future Web as they are rapidly being integrated into our lives. Smart homes operating

with Web technologies could be adapted with little effort, to support interaction with social networking sites, for sharing the status of the house as well as individual home devices/services, between family members and other tenants.

Online social networking could be also utilized for establishing energy competitions between neighboring houses and flats. In addition, it could be used for electricity-related comparisons, to help people understand their electricity footprint, i.e. the amount of electrical energy they consume. In such applications, social norms have the potential to motivate people to acquire more sustainable behaviors [12]. Residents in general tend to learn from their neighbors and friends, and receive encouragement and support.

Moreover, increasing energy demands and environmental concerns influenced initiatives towards a more rational utilization of electrical energy. Thus, traditional electricity grids are planned to be converted into modern smart grids. A smart grid represents the future electricity grid, enhanced with Information and Communication Technology (ICT), applied to generation, delivery and consumption of electric power. The smart grid vision requires near real-time communication and synchronization with energy-aware smart homes, to establish its goals for effective and efficient energy management and distribution. To enable this vision, a common ground needs to be specified, a common language understood by all smart homes, facilitating in-home and home-to-grid communication. Also in this case, the Web, as a scalable, pervasive and flexible platform, could be an appropriate solution for such a wide-scale interconnection.

Finally, in order to enable the vision of interoperable smart homes operating with Web technologies, reliability and performance need to be ensured. Especially when considering specific smart home applications dealing with safety, health, advanced automation and energy-related programs of the smart grid, reliable message exchange between home devices in satisfactory amounts

of time must be guaranteed. These guarantees may be obtained by employing intermediate request queues, installed as mediators between the smart home application and the physical devices. Shaman [148] was an early gateway system that included request queues for the communication with resource-constrained devices. This gateway system partly motivated this work, in regard to the request queue mechanism (see Chapter 6), integrated inside an application framework for smart homes (see Chapter 4). The analysis provided helps in the selection of the appropriate design parameters for the request queue mechanism, such as the retransmission interval timer.

1.3 Reasoning about Web of Things and REST

The great success of the Web as a global platform for information sharing educates that simple and loosely-coupled approaches offer a high degree of scalability, robustness and evolvability. This success is partly due to an open and uniform interface, which enables non-experts to develop interactive applications rapidly. Unlike most protocols used in embedded computing, Web standards are widely used in the Internet, being highly flexible.

Using a Web-based approach for home automation offers several benefits, such as wide-spread deployment of Web browsers and ubiquitous HTTP support in programming and scripting languages. Residents can employ any computing device to manage their smart home (e.g. mobile phones, tablets, smart TVs, PCs, home routers, smart meters), provided these devices has access to the Internet/Web. Open access to home devices and their environmental context through Web services, using open and simple standards (e.g. XML, HTTP, JSON) enables information to be reused across independent systems, lowering the access barrier that allows people to develop their own composite applications. People may combine ubiquitous services towards home automation following the classical Web mashup techniques, enabling the Internet/Web to reach out into the real world of smart homes.

REST is considered a suitable architectural style to enable the communication with home residents and physical devices. It can guarantee interoperability and a smooth transition from the Web to home environments. A resource-oriented environment facilitates the residents' efforts in regard to configuring their smart homes. Tasks that include browsing for home devices, linking them together, searching for their services and interacting with them, combining their functionality, become easier and more standardized through resource-oriented architectures.

Undoubtedly, many standards offer interoperability between heterogeneous machines. Their main difference from REST is that they include a rather complicated set of protocols and guidelines for interaction between devices, while REST suggests to use merely the HTTP protocol and basic Web principles to address heterogeneity. A recent study [73] compares REST with its main opponent, Big Web services (WS-*) [13]. The study suggests that REST is easier to learn and understand, more lightweight, flexible and scalable, better for heterogeneous environments while WS-* has better security features, but is more complex to be used, being more appropriate for business applications. The validity of this comparison is reinforced by the work in [130].

Although WS-* is in general inappropriate for pervasive computing, recently the WS-* community introduced the Devices Profile for Web Services (DPWS)[2], defined as a minimal set of implementation constraints to enable secure WS-* messaging, discovery, description, and eventing on resource-constrained devices. DPWS is aligned with Web services technology and includes numerous extension points allowing for seamless integration of device-provided services in enterprise-wide application scenarios. Related to DPWS, the Web Services for Devices (WS4D)[3] is a recent initiative, bringing WS-* technology to the application domains of industrial automation, home entertainment, automotive systems and telecommunication systems. WS4D offers

[2]http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01

[3]http://ws4d.e-technik.uni-rostock.de/

toolkits that comply with the DPWS protocol and a software tool to create device mashups called WS4D-PipesBox [19]. The fact that WS-* community realized the need to support the world of embedded devices towards interoperability is of course very positive. A comparison between REST and DPWS/WS4D would be an interesting task, as future work. Still, we prefer REST as it models the actual operation of the Web in ubiquitous scenarios such as smart homes, while WS-*-based protocols use the Web merely as the transport protocol.

Extensible Messaging and Presence Protocol (XMPP)[4] is an open XML-based communications technology that could be applied, however, it is heavy for use with embedded devices. JXTA [158] is another alternative, but it is more of an overlay network, not directly related to the concept of the Web. CoAP [150] is a tailored implementation of REST for constrained environments, which could be considered in the future when it possibly becomes more mature.

1.4 Contribution

In this thesis, the main contribution is the development of an application framework for smart homes and the presentation of an inclusive approach for developing smart home applications based on reusing Web standards. Even though a number of energy-related applications are discussed, we do not present a novel technology for energy efficiency. The main innovation lies in the fact that existing Web technologies are combined to develop a generic smart home application framework that can be easily adapted to future technological and societal challenges.

To address heterogeneity of home devices and smart home technologies, this work examines whether the Web, as a highly ubiquitous, reliable and scalable platform, can constitute an effective and efficient application layer for future smart home solutions. As the Web is pervasive in our everyday lives, with well-trusted and understood protocols and techniques, its potential utilization is

[4] http://xmpp.org/

investigated, for achieving interoperability in the home environment. A number of representative applications are developed to exemplify the power of the proposed framework and approach.

Specifically, the applicability of employing Web technologies in the smart home environment is examined from various aspects:

- The development of a RESTful application framework for smart homes using Web techniques (see Chapter 4).
- The issues encountered during the Web-enablement of home devices, including device discovery, service description and uniform interaction with them. (see Chapter 5).
- A technical evaluation showing that the application of Web technologies in recourse-constrained home environments may offer satisfactory performance, in terms of response times and battery lifetime of the embedded devices of the smart home (see Chapter 7).
- Online social networking of the smart home, for sharing home devices between family members and for performing comparisons of electrical consumption between neighbors and friends (see Chapter 8).
- Integrating energy-aware smart homes to the smart grid of electricity (see Chapter 9).
- Considering technological and societal challenges beyond the smart home (see Chapter 10).

Our research study is unique -to our knowledge- in the sense that it explores the applicability of the WoT in a real-life scenario, such as a smart home environment. Most of the previous studies in the domain of the WoT [159, 72] focused on defining the foundational blocks of a Web of physical things. In this work, the focus is to propose a flexible application framework, and to solve numerous issues that appear when applying the WoT in a smart home environment, towards

achieving flexibility and interoperability between heterogeneous home devices and services, in order to ensure acceptable performance inside this wireless indoor space.

A generic application framework for smart homes was developed, supporting the management of Web-enabled physical devices and the efficient interaction with multiple home residents, remotely through the Web (see Chapter 4). This framework bridges heterogeneous devices and the Web, regardless of the actual communication protocol used natively by the devices. The layered design of the framework, integrating threads and request queues for managing the communication with physical devices, allows effective handling of failures, which are possible in the embedded environment of home devices. In case transmission failures occur, fast retransmissions ensure reliability and acceptable performance.

The use of request queues as a mechanism to administer the interaction with embedded devices constitutes a novel characteristic in smart home middleware (see Chapter 6). Request queues offer enhanced reliability and fault tolerance, supporting multiple tenants simultaneously as well as prioritized requests. Load balancing is used to increase the battery lifetime of devices while concepts from queuing theory allow better designs of the queues and the setting of design parameters, and offer approximate estimations about the time needed for satisfying incoming requests.

The application framework presents a unique architecture, based on resource-oriented architectures and REST, for discovering, searching and interacting with heterogeneous embedded devices over the Web. This capability is useful, but not sufficient to develop complete applications for end users. Hence, the framework supports smart application development over REST, following the physical mashup paradigm (see Section 4.2.6), enabling home residents to easily access, program and integrate devices and environmental data into their own Web applications. Residents can manually implement their own small programs in any programming language that supports

HTTP (e.g. Java, Python, PHP, AJAX), in the form of physical mashups. Physical mashup programming is a concept first mentioned in [75], which we applied specifically for smart home environments. We adapted physical mashups for highly-dynamic and mobile urban environments, proposing urban mashups (see Section 4.2.7), defined as opportunistic service-centric physical mashups, validated only when the local environmental conditions support the sensor-based Web services declared by these mashups.

The application framework was then extended into a Web-based generic, graphical application for smart homes, where all interactions with embedded devices are done via standard HTTP requests (see Section 4.3). Since many residents are not expected to have any experience in programming, we also give them the opportunity to develop their own smart rules through a physical mashups editor (see Section 4.3.4). Our aim is to bring the development of home automation applications closer to Web developers, but also to facilitate the creation of simple applications by end users, tailored to their needs.

Furthermore, a 6LoWPAN-based WSN was deployed in a real environment and home devices were transformed into embedded Web servers, exposing their capabilities as RESTful Web services (see Chapter 5). This effort was also performed, in parallel to this work, in [182, 147]. It is a common secret that porting the TCP/IP stack on resource-constrained devices creates degradation of their overall performance. However, our evaluation efforts (see Chapter 7) indicate that the employment of Web techniques can mitigate or even cancel this performance penalty. More specifically, enhanced Web techniques such as HTTP caching and event-based Web messaging can contribute in increasing the performance and scalability of smart applications, reducing the response time and power consumption of sensor devices, preserving their battery lifetime. These findings are missing from related work.

Moreover, our contribution involves a number of case studies for demonstrating how the Web behavior of smart homes can be exploited for facing environmental, societal and energy-related issues, which are encountered in the modern cities where the majority of people lives. We demonstrate, through the implementation of various pervasive applications, how flexible and easy smart application development becomes, when using the Web as an application protocol.

The integration of online social networking to Web-enabled smart homes provides a social shape to smart home applications (see Chapter 8). By means of the mechanisms offered by social networking sites, family members can share the status of their smart home and allow shared interactions with home devices and their services (see Section 8.1).

Social influence, as an important parameter for energy awareness and conservation, was considered in a social competition for energy conservation between neighboring flats in large blocks of flats (see Section 8.2). The findings of the competition were encouraging, because people decreased their electrical consumption significantly during the testing period.

Social Electricity is a promising, novel Facebook application that permits people to compare their electricity footprint with their friends, to understand whether they consume more electricity than they should (see Section 8.3). This application has been awarded the first prize in an international competition for sustainable applications (2nd ITU Green ICT Application Challenge), organized by the International Telecommunications Union (ITU), which is the United Nations specialized agency for ICT protocols and technologies. Social Electricity is currently being deployed around Cyprus and hundreds of Cypriot citizens are using it to perceive the ”semantics” of their consumed energy.

The possibilities created when Web-based, energy-aware smart homes communicate in near real-time with the smart grid are examined in Chapter 9, and an architecture for the flexible integration of smart homes to the grid through the Web is proposed (see Section 9.1). The potential of this

Web-based architecture is demonstrated through the development of two applications that exploit these new capabilities of smart homes, towards an intelligent grid. Demand response is harnessed to schedule electricity-related tasks for future execution (see Section 9.2) and load shedding is employed to reduce the total load for avoiding outages (see Section 9.3).

Finally, our contribution is extended to some issues beyond the smart home environment. The future (expected) abundance of Web-enabled sensors would imply the need for efficient techniques for globally discovering them. To solve this problem using the existing Internet infrastructure, we explore the exploitation of the Domain Name System (DNS) as a scalable, ubiquitous, real-time directory mechanism for embedded devices (see Section 10.3).

Web-enabled home devices of future smart homes, mainly those offering sensing capabilities, could be shared through the Web under a participatory-based crowdsourcing scheme, for automatic knowledge retrieval in the urban environment (see Section 10.4). UrbanRadar is a mobile application we developed, which uses urban mashups as a core element for location-based environmental awareness (see Section 10.4.2). Through this case study, we aim to promote the important role of Web-enabled smart homes towards shaping the digital, real-time future cities.

1.5 Acknowledgements of Contributions

Several people took part in realizing the abovementioned scientific contributions, to whom I am grateful for their cooperation and guidance.

At first, I acknowledge the significant contribution of my thesis advisor Dr. Andreas Pitsillides, Professor at the Computer Science department of the University of Cyprus, who monitored closely my progress throughout this thesis, giving me valuable advice and feedback.

This thesis has been performed in parallel to the work of Dr. Vlad Trifa [159]. We have collaborated with Dr. Trifa in many aspects of this work, as well as of his own PhD thesis. Thus,

parts of the work presented here is complementary in content - yet compatible in concept - with his dissertation. Additionally, this thesis is influenced by the work of Dr. Dominique Guinard [72]. References to the related work of Dr. Guinard and Dr. Trifa are given throughout the thesis.

In particular, Dr. Trifa, co-founder and chief product officer of a startup company called Evrythng[5] for motivating the Web of Things as the basic notion behind the development of a Web-based application framework for smart homes, in Section 4.2. His work in event-based Web messaging inspired the integration of push techniques in home sensor devices, for triggering fast notifications through Web messaging in case of sporadic events (see Section 5.2.4).

Dr. Trifa, as well as Dr. Guinard, CTO and co-founder of Evrythng company, influenced this work in terms of adopting physical mashups for combining real-world services offered by household devices with Web resources (see Section 4.2.6).

Yiannis Tofis, Ph.D. student at KIOS research center, has contributed to the work presented in Section 9.3, with his knowledge and experience in the smart grid. He helped in simulating a smart grid controller, to enable the development of a load shedding scenario. I also acknowledge Dr. Chakib Bekara, Post Doc fellow at the Fraunhofer Fokus Institute, for sharing with me his knowledge in securing smart homes that operate with Web technologies, in Section 10.5.

Finally, I would like to acknowledge the work of some undergraduate students of the department of Computer Science at the University of Cyprus, who contributed mainly in terms of implementing specific aspects of the thesis, during their bachelor dissertation. Michalis Yiallouros, who implemented part of the graphical interface for Web-based smart homes in Section 4.3, Giannis Kitromilides, for his efforts in achieving the social competition for energy conservation in blocks of flats (see Section 8.2), Diomidis Papadiomidous, who developed an early version of Social Electricity Facebook application (see Section 8.3.1), George Taliadoros for improving this

[5] http://evrythng.net

social application and enhancing it with advanced functionalities and Marios Michael, who helped in deploying Social Electricity around Cyprus and performing an evaluation of the project among the Cypriot citizens.

1.6 Publications

A list of publications stemming from the work in this thesis, is provided in Appendix A.

1.7 Thesis Overview

This thesis is organized as follows: Chapter 2 presents general background information, necessary in order to understand the concepts presented in this work. This information covers embedded sensor technology, the IoT and the WoT, as well as REST and resource-oriented architectures. It also covers protocols and techniques used on the Web for event-based messaging and for message exchange using standardized data formats.

Chapter 3 lists the important related work in the field, including state of the art in ubiquitous middleware, smart homes, smart metering, the IoT, the WoT and large-scale sensor platforms. Special focus is given on middleware and smart home solutions that integrate Web technologies.

After, Chapter 4 describes a Web-based application framework for smart homes, designed using Web technologies, in order to address the problem of heterogeneity in home environments. Requirements for future smart homes are listed and the general architecture of the framework is discussed. A graphical application for smart homes, built on top of the application framework is presented, and various case studies are provided, based on the physical mashup paradigm.

Chapter 5 presents the issues encountered during the process of Web-enabling the embedded device of a smart home. Various aspects are addressed, including the discovery of the devices, the description of their services, the interaction patterns with them etc.

Then, Chapter 6 analyzes the behavior of the request queue mechanism, focusing on the fast retransmission functionality. Potential benefits of employing intermediate request queues are discussed and various of these benefits are demonstrated experimentally.

Chapter 7 depicts the use of Web techniques for enhancing the performance of home devices, mainly in terms of their response times and their energy consumption. Event handling and a streaming network are two scenarios also evaluated in this chapter.

Afterwards, Chapter 8 presents some pervasive applications, enabled when combining Web-enabled smart homes and online social networking sites. These applications include: an application for sharing home devices and services between family members; a social competition for energy conservation in blocks of flats; and a Facebook application for comparison of electrical consumption between friends.

Chapter 9 considers the integration of energy-aware smart homes to the smart grid of electricity by means of the Web. An architecture is proposed for this interconnection and two case studies are considered. The first exploits the demand response program of the grid to execute energy-related tasks in low-tariff hours of the day and the second employs load shedding for reducing the total load of the grid in case outages are possible to occur.

Chapter 10 introduces research issues which are beyond smart home environments. These issues involve extending the application framework for other real-life scenarios, a real-time, global search engine for the physical world as well as advanced knowledge retrieval in urban environments. This chapter discusses also some basic security aspects both inside the smart home and between the house and other Web entities.

Finally, Chapter 11 concludes the thesis by discussing the general findings, identifying open issues, giving a general outlook of the future potential of Web-enabled smart homes and proposing future work in the field.

Chapter 2

Background Information

In this chapter, we provide some background information to help the reader understand the concepts and the technologies discussed in this proposal. At first, Section 2.1 provides general information about embedded sensor technology. More precisely, Sections 2.1.1 and 2.1.2 give a description of the embedded sensor and actuator devices we employ for building a smart home, namely sensor motes, residential smart meters and smart power outlets. Afterwards, Section 2.1.3 discusses the formation of wireless sensor networks, by exploiting the wireless capabilities of embedded devices, while Section 2.1.4 presents some well-known operating systems for sensor motes, namely TinyOS and Contiki. Then, Sections 2.1.5 and 2.1.6 explain IEEE 802.15.4, ZigBee and 6LoWPAN, which are popular standards for low-power wireless communications. Section 2.2 introduces the concept of the Internet of Things and Section 2.3 shows how this concept can be extended for establishing a Web of Things. After, Section 2.4 gives an overview of resource-oriented architectures and REST, which constitute an important element for building a true Web of physical devices. Section 2.5 provides well-known data formats that are used for message exchange on the Web, as well as MIME types, as a popular standard that describes the structure of messages on the Internet, and Section 2.6 explains the available protocols for push/pull messaging

on the Web. Finally, Section 2.7 lists specifically the various technologies adopted at each chapter of the thesis.

2.1 Embedded Sensor Technology

This section provides an overview of the technological advancements in embedded sensor technology. The section begins with a description of embedded sensor devices, residential smart meters and smart power outlets, and continues with defining wireless networks of sensor and actuator devices. Finally, various wireless protocols for resource-constrained environments are discussed, involving IEEE 802.15.4, ZigBee and 6LoWPAN.

2.1.1 Sensor Devices

Recent advances in micro-electro-mechanical systems technology, wireless communications, and digital electronics have enabled the development of low-cost, low-power, multi-functional sensor nodes that are small in size and communicate untethered in short distances [11].

Sensors can measure with high accuracy environmental conditions such as temperature and humidity or physical events such as pressure and motion. The reader can graphically see such a sensor device (Tmote Sky sensor mote) in the left of Figure 2.1. In the right of the same figure, the internal design of this sensor platform is presented. Tmote Sky motes have the same architecture and design as Telosb motes [134].

Such platforms are ideal for low power, relatively high data-rate sensor network applications designed with the dual goal of fault tolerance and development ease. They boast a large on-chip RAM size, IEEE 802.15.4 radio and an integrated on-board antenna, providing range in theory up to 125 meter. However, in practice, when obstacles exist such as walls or trees, their radius reaches only 20-30 meters. This is also the case for indoor environments such as smart homes,

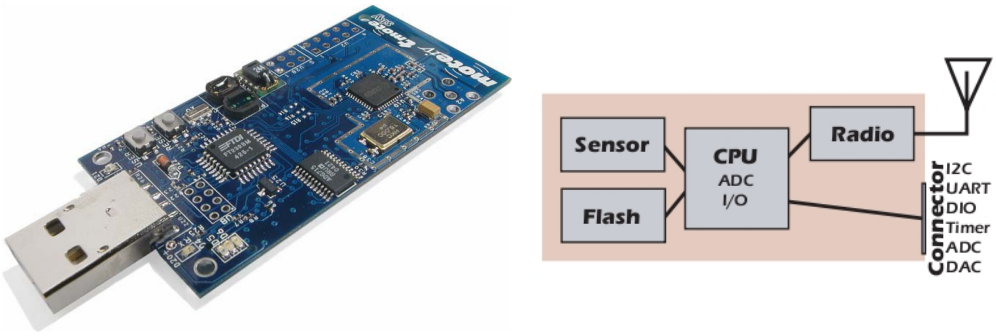

Figure 2.1: A Tmote Sky physical sensor device and its internal design.

as it is shown in our analysis and evaluation efforts in Chapters 6 and 7. Sensor motes also offer a number of integrated peripherals, including ADC and DAC, timers, I2C, SPI and UART bus protocols, as well as performance-boosting DMA controllers.

2.1.2 Residential Smart Meters and Smart Power Outlets

In the latest years, residential smart meters have gained popularity. Electrical smart meters measure the consumption of electrical energy in houses and buildings, in frequent intervals and communicate that information at least daily back to the utility for monitoring and billing purposes. Smart meters are a fundamental element of the future smart grid of electricity. However, smart metering does not only affect the future development of the smart grid, but also motivates the rational management of the electrical consumption in houses and buildings. It is planned that every home in Britain will be equipped with smart meters by 2020[1].

Smart power outlets are wireless devices that measure the consumption of individual electrical appliances and control their operation in real-time. They may form multi-hop wireless networks inside the home environment, to propagate the electricity-related measurements to a central smart home application. Another category of smart devices, which are associated with electrical consumption are smart appliances. These devices are an important element in realizing the benefits

[1] http://news.bbc.co.uk/2/hi/business/8042716.stm

of smart grid technologies. They have the potential to significantly improve the stability and operational efficiency of the electrical grid with limited impact on the lives of the residents. Their main operation involves responding to pricing signals from the grid, and decide when it is most economical to execute their task.

According to various scientific studies [37, 179], timely electrical consumption feedback through smart metering, is believed to reduce electrical consumption by a fraction of 5-15%. Enabling residents to analyze and monitor their electricity footprint in real-time using intuitive visualization tools, allows intelligent and efficient energy management. In some cases, analytical feedback (per room/electrical appliance) can reduce energy consumption up to 20% [138].

The American Council for an Energy-Efficient Economy (ACEEE) reviewed more than 36 different residential smart metering and feedback programmes internationally. This is one of the most extensive studies of its kind. Their conclusion was that "to realise potential feedback-induced savings, smart meters must be used in conjunction with in-home displays and well-designed programmes that successfully inform, engage, empower and motivate people" [97].

There are near universal calls from both the energy industry and consumer groups for a national social marketing campaign to help raise awareness of smart metering and give customers the information and support they need to become more energy efficient, and what changes they must make to realize the potential of proposed smart meters.

Residential smart meters are placed where the home connects to the power grid, measuring the total energy consumption of the house. Figure 2.2 (left) presents a typical residential smart meter, installed on the main supply of the house. Figure 2.2 (right) shows a commercial smart power outlet. These outlets need to be plugged on electrical appliances in order to measure their electrical consumption and manage their operation.

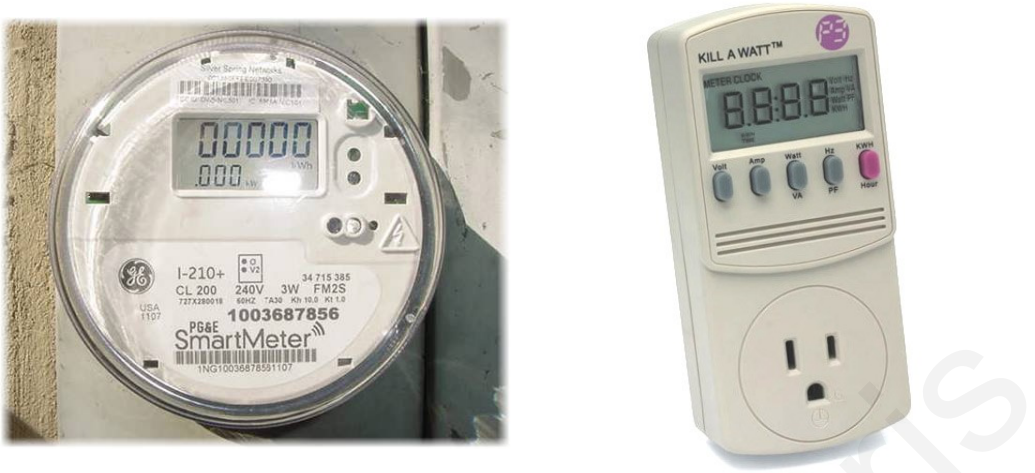

Figure 2.2: A typical residential smart meter (left) and a smart power outlet (right).

Whole-home residential smart metering products available in the market include Wattson[2], Onzo[3] and Current Cost[4]. Device-specific smart power outlets include Kill A Watt[5] and Ploggs[6]. A comparison between electrical smart meters and smart power outlets, considering their overall accuracy is presented in [111].

2.1.3 Wireless Sensor Networks

A WSN consists of spatially distributed autonomous sensors, which cooperatively monitor physical or environmental conditions, such as temperature, humidity, sound, vibration, pressure, motion, pollutants etc. [11].

Each sensor runs a multi-hop routing algorithm, where intermediary nodes function as forwarders, relaying data packets to a base station (sink). Base stations have much more computational, energy and communication resources and act as gateways between sensor nodes and the

[2]http://www.diykyoto.com/uk

[3]http://onzo.com/

[4]http://www.currentcost.com/

[5]http://www.p3international.com/products/special/p4400/p4400-ce.html

[6]http://www.plogg.co.uk/

end users. The topology of WSN can vary from a simple star network to an advanced multi-hop wireless mesh network. The propagation technique between the hops of the network can be routing or flooding.

The main characteristics of WSN include: power consumption constraints for nodes using batteries or energy harvesting; ability to cope with node and communication failures; node mobility; heterogeneity of nodes; scalability to large deployments; ability to withstand harsh environmental conditions; and ease of use.

WSN play a key role in enabling high-precision sensor and actuation systems in houses, buildings and surrounding spaces by providing a reliable, cost-effective and extensible solution. Their equipment can be placed in existing as well as new structures, without significant changes in the current infrastructure. Common applications of WSN involve forest fire detection, air pollution monitoring, health monitoring, industrial control, agriculture and greenhouse monitoring. WSN can also be deployed in inaccessible rural areas where human transition is difficult e.g. forests, jungles and volcano.

In Figure 2.3, we can observe a deployment of a WSN on an active volcano [177]. In this particular deployment, wireless sensor nodes detect seismic events and route them, through the wireless network to a base station.

2.1.4 Operating Systems for Sensor Devices

A number of operating systems for embedded devices and especially sensor motes have been developed during the last 10-15 years. The most well-known are TinyOS and Contiki.

TinyOS [110] is a free and open-source, component-based operating system and platform, targeting WSN. It is an embedded operating system, written in the nesC programming language as a set of cooperating tasks and processes. It basically started as a collaboration between the

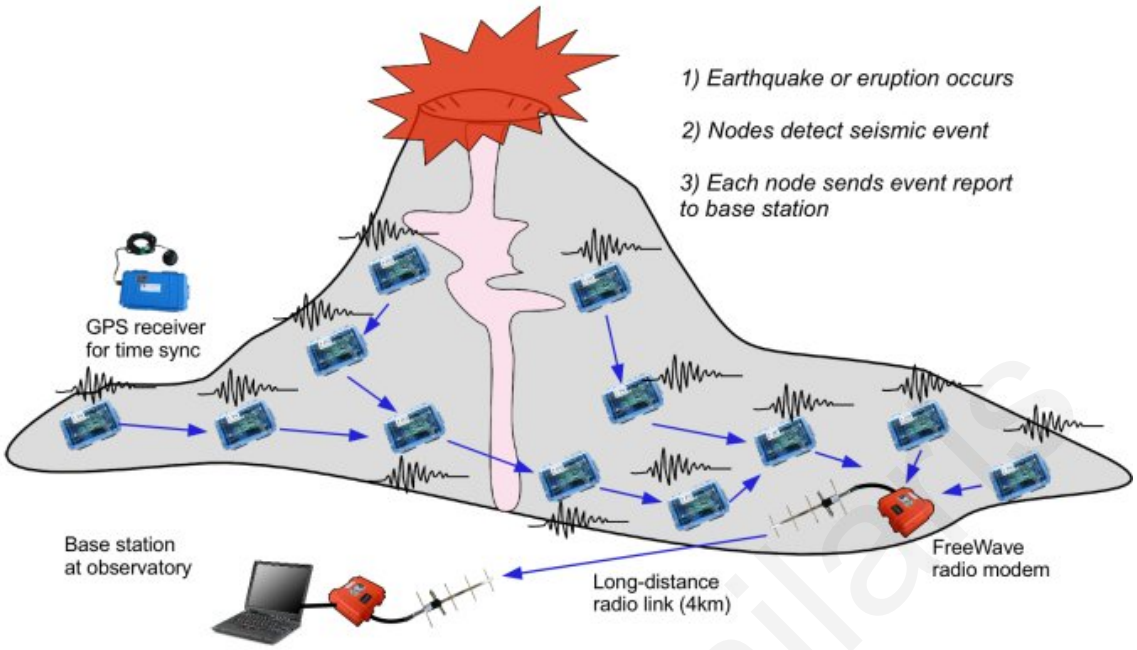

Figure 2.3: A deployment of a WSN for volcano monitoring [177].

University of California, Berkeley and Intel Research, and has since grown to a be an international consortium, the TinyOS Alliance.

TinyOS applications are written in nesC, which is actually a dialect of the C programming language, optimized for the memory limitations of sensor networks. TinyOS programs are built out of software components, which are connected to each other using interfaces. Interfaces contain definitions of commands and events. Components must implement the events they use and the commands they provide. TinyOS provides interfaces and components for common hardware abstractions such as packet communication, routing, sensing, actuation and storage. A general idea of the programming model can be seen in Figure 2.4.

TinyOS is completely non-blocking: it has one stack. Therefore, all I/O operations that last longer than a few hundred microseconds are asynchronous and have a callback. TinyOS uses nesC's features to link these callbacks, called events, statically. A TinyOS component can post a

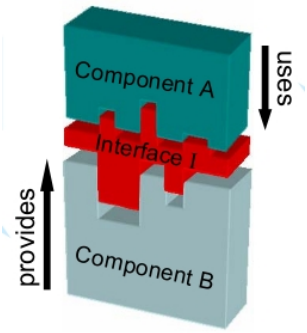

Figure 2.4: General programming model of TinyOS.

task, which the OS will schedule to run later. Tasks are non-preemptive and run in first in, first out (FIFO) order. This simple concurrency model is typically sufficient for I/O-centric applications, but its difficulty with CPU-heavy applications led to the development of TOSThreads, a thread library for the OS.

Contiki [47] is also an open-source, highly portable, multitasking operating system for memory-efficient networked embedded systems and wireless sensor networks. It is mainly designed for microcontrollers with small amounts of memory. A typical Contiki configuration is 2 kilobytes of RAM and 40 kilobytes of ROM. For communication within a wireless sensor network, Contiki uses the RIME [46] low-power radio networking stack. The RIME stack implements sensor network protocols ranging from reliable data collection and best-effort network flooding to multi-hop bulk data transfer and data dissemination.

Contiki is written in the C programming language and consists of an event-driven kernel, on top of which application programs can be dynamically loaded and unloaded at run time. Contiki processes use lightweight protothreads [48] that provide a linear, thread-like programming style on top of the event-driven kernel. In addition to protothreads, Contiki also supports per-process optional multithreading and interprocess communication using message passing.

To provide a long sensor network lifetime, it is crucial to control and reduce the power consumption of each sensor node. Contiki provides a software-based power profiling mechanism that keeps track of the energy expenditure of each sensor node. Being software-based, the mechanism allows power profiling at the network scale without any additional hardware. Contiki's power profiling mechanism is used both as a research tool for experimental evaluation of sensor network protocols and as a way to estimate the lifetime of a network of sensors.

A full installation of Contiki includes the following features: a multitasking kernel; optional per-application pre-emptive multithreading; protothreads; TCP/IP networking, including IPv6; Windowing system and GUI; networked remote display using Virtual Network Computing; a small Web browser; a personal web server; and a simple telnet client.

Other, less popular operating systems for embedded devices include MANTIS [21], Nano-RK [2] and LiteOS [1].

2.1.5 IEEE 802.15.4 and ZigBee

IEEE 802.15.4 [54] is the proposed standard for low-rate wireless personal area networks. It defines the physical and media access control layer of these networks (based on the OSI model). It operates in three frequency bands as shown in Table 2.1.

IEEE 802.15.4 is designed for wireless sensor applications that require short-range communications, to maximize the battery lifetime of sensor devices. The main features of the standard include activation and deactivation of the radio transceiver, energy detection within the current channel, link quality indication for received packets, channel selection, clear channel assessment (CCA), and transmitting as well as receiving packets across the physical medium. Other important features include real-time suitability by reservation of guaranteed time slots, collision avoidance through CSMA/CA and integrated support for secure communications.

ZigBee [36, 54] is a specification for a suite of high-level communication protocols using small, low-power digital radios based on the IEEE 802.15.4 standard for personal area networks. ZigBee technology offers low data rates, low power consumption and low cost, proposing a wireless networking protocol targeted for automation and remote control applications. ZigBee builds upon the physical layer and medium access control defined in IEEE802.15.4 standard (2003 version). The specification completes the standard by adding four main components: network layer; application layer; ZigBee device objects (ZDOs); and manufacturer-defined application objects which allow for customization and integration.

IEEE 802.15.4 committee started working on a low data rate standard some decades ago. Then, the ZigBee Alliance and the IEEE decided to join forces and ZigBee is the commercial name for this technology. ZigBee targets radio-frequency (RF) applications that require long battery lifetime, low data rate and secure networking. It operates in the industrial, scientific and medical (ISM) radio bands: 868 MHz in Europe, 915 MHz in the USA and Australia, and 2.4 GHz worldwide. Data transmission rates vary from 20 to 900 kilobits/second.

ZigBee has been developed to meet the growing demand for robust multi-hop wireless networking between numerous low-power devices. ZigBee devices are often used in a mesh networking topology, to transmit data over longer distances, passing them through intermediate devices to reach more distant ones. This allows ZigBee networks to be formed in a decentralized, ad-hoc manner, without centralized control.

In the industry, ZigBee is being used for next generation automated manufacturing, embedding small transmitters in large numbers of embedded devices, in order to exchange wirelessly sensory observations in a multi-hop fashion. Lately, ZigBee has developed standards for specific applications such as home and building automation, and the smart grid.

Table 2.1: 802.15.4 physical layer.

Frequency (MHz)	**Modulation**	**Channels No.**	**Bit rate(Kbps)**
868-868.6	BPSK	1	20
902-928	BPSK	10	40
2400-2483.5	16-ary QPSK	16	250

Other standards based on IEEE 802.15.4 are WirelessHART and 6LoWPAN. WirelessHART [5] is intended for low power, 2.4 GHz operation, expecting a huge need for wireless instrumentation. WirelessHART is compatible with all existing devices, tools and systems. WirelessHART is reliable, secure, and energy-efficient, supporting mesh networking, channel hopping, and time-synchronized messaging. Network communication is secure with encryption, verification, authentication, and key management. 6LoWPAN is an adaption layer that allows efficient IPv6 communication over IEEE 802.15.4. We will further discuss 6LoWPAN in the next subsection.

2.1.6 6LoWPAN

Integrating IP with embedded devices and WSN has a number of important benefits. Those benefits include the easy interconnection with other IP networks, the reuse of the existing Internet infrastructure, the application of well-known IP-based technologies in ubiquitous environments and the use of existing power monitoring and diagnostic tools. The challenge in supporting IP protocols in WSN is to overcome the limitations experienced by sensor networks like lower power consumption, low duty cycles, limited bandwidth and reduced reliability.

To address these limitations, IETF created the 6LoWPAN working group, which aims to develop the support of IPv6 over the standard IEEE 802.15.4, in order to import well-known capabilities of IPv6, such as Neighbor Discovery (ND) and Mobile IP (MIPv6) into low-power devices. A

Low-power Wireless Personal Area Network (LoWPAN) is a simple, low-cost communication network that allows wireless connectivity in applications with limited power and relaxed throughput requirements. A LoWPAN typically includes devices that work together to connect the physical environment to real-world applications.

IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) [119, 107], as stated before, is an adaption layer that allows efficient IPv6 communication over IEEE 802.15.4. It defines a LoWPAN frame format for IPv6 data packets and a simple header compression scheme which uses shared context information. Also, mesh-based packet delivery is briefly addressed.

The most well-known operating systems for sensors offer already 6LoWPAN implementations, for transforming sensor motes into IPv6-enabled devices. These implementations are blip for TinyOS and uIPv6 for Contiki.

Blip 2.0 [38], the Berkeley low-power IP stack, is an implementation in TinyOS of a number of IP-based protocols. Using blip/TinyOS, multi-hop IP networks may be formed, consisting of different motes communicating over shared protocols.

uIPv6 [50] is an open-source TCP/IP stack, capable of being used with tiny 8- and 16-bit microcontrollers. It was initially developed by Adam Dunkels of the Networked Embedded Systems group at the Swedish Institute of Computer Science, initially called uIP. In October 2008, Cisco, Atmel and SICS announced a fully compliant IPv6 extension to uIP, called uIPv6

A comparison of IPv6/6LoWPAN and ZigBee is provided in Table 2.2. Obviously, 6LoWPAN offers more capabilities than ZigBee, supporting a larger number of sensor devices, better data rates, low cost and higher connectivity.

Table 2.2: Comparison of IPv6/6LoWPAN and ZigBee.

	ZigBee	**IPv6/6LoWPAN**
Network size	2^{16}	2^{64} per subnet
Data rate	20..250kb/s	250kb/s..1Gb/s
Interface	App. gateway	UDP, TCP, RESTful Web
Maturity	2004	1998
Costs	medium	low
Installation overhead	low	low
Connectivity	medium	high
Security	medium (AES)	medium (AES)

2.2 The Internet of Things

New technologies like short-range wireless communications, RFID and real-time localization are now becoming largely common, allowing the Internet to penetrate into the real world of physical objects. The introduction of IPv6 [39, 143], the efforts for porting the IP stack on embedded devices [49, 86] and the definition of 6LoWPAN (see Section 2.1.6) enable the vision of the *Internet of Things* [62, 115], which refers to a network of objects, where all things are uniquely and universally addressable, identified and managed by computers. It is a collection of technologies that make it possible to connect things like sensors and actuators to the Internet.

A formal definition of the IoT is the following: "The Internet of Things is an integrated part of the Future Internet and could be defined as a dynamic global network infrastructure with self configuring capabilities based on standard and interoperable communication protocols where physical

and virtual things have identities, physical attributes, and virtual personalities and use intelligent interfaces, and are seamlessly integrated into the information network.[7]"

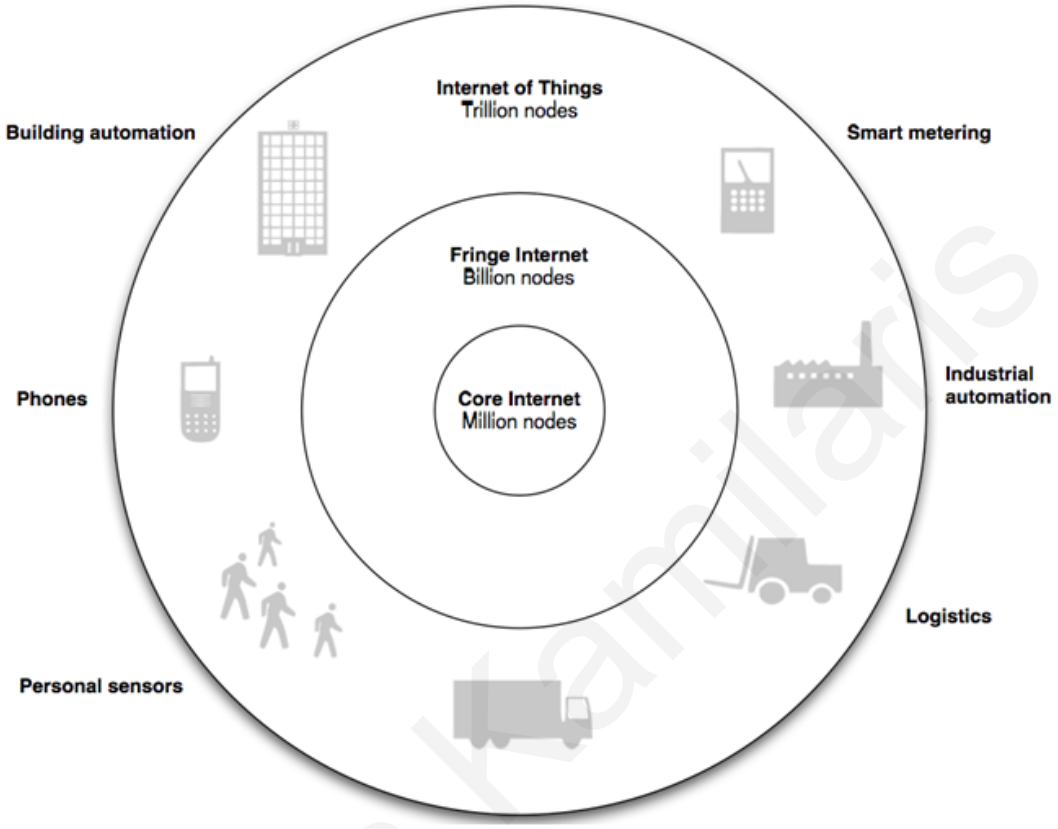

Figure 2.5: The Internet Of Things vision.

The IoT aims at extending current Internet, which includes billions on nodes, into the everyday world, employing trillions of Internet-enabled nodes. These nodes include sensors, actuators, smart meters, smart power outlets, information appliances, RFID tags etc. These embedded devices will be harnessed in a broad range of applications, such as building and industrial automation, smart metering, logistics etc. The IoT vision is depicted in Figure 2.5.

[7] http://www.internet-of-things.eu/

The specific characteristics of the IoT are still under research. Preferences such as its architecture, size, complexity, time and space issues, as well as the ambient intelligence and autonomous control of this novel concept are still in a premature level.

Current applications of the IoT are found in logistics (RFID tags), smart homes (ZigBee/6LoWPAN), large-scale platforms for sensor data (Cosm [82], Paraimpu[8]), in business cards (QR codes) etc.

2.3 The Web of Things

Extending the concept of the IoT, the *Web of Things* [178, 76] is a notion where everyday devices and sensors are connected by fully integrating them to the Web, as shown in Figure 2.6. Based on the success of the Web 2.0, this concept is about reusing well-accepted and understood Web standards to connect constrained devices.

Figure 2.6: The Web Of Things vision.

[8] http://paraimpu.crs4.it/

The WoT introduces many benefits in the ubiquitous computing society. It brings a new application development paradigm, where applications can be easily built on top of embedded devices, utilizing the same techniques used in Web development, for example Web mashups that involve physical things (see Sections 4.2.6 and 4.2.7). Furthermore, many features of the Web can by employed very easily. These features include searching, securing, blogging, caching, linking, bookmarking etc. Finally, the Web has proved to be a highly scalable and ubiquitous platform, and researchers believe it is appropriate for interconnecting millions of embedded devices and everyday objects under uniform and interoperable interfaces.

The WoT practice mainly follows three steps:

1. Connect embedded devices to the Internet, through IPv4 or IPv6 (see Section 2.1.6).
2. Embed Web servers on these devices.
3. Model the services, offered by these devices, in a resource-oriented way (see Section 2.4).
4. Expose these services as RESTful Web resources (see Section 5.2.3).

Actually, the Web-enabling process can be performed in two ways. Either through embedding Web servers directly on physical devices (indicated way) or by employing gateways, which perform protocol translation, from the TCP/IP protocol to the dedicated protocol used by the physical device (e.g. Bluetooth, ZigBee). Gateways are used when it is not possible to integrate the TCP/IP protocol stack to embedded devices (e.g. RFID tags, QR codes). Smart meters and smart power outlets do not yet support the direct embedding of Web servers on them, however, this is expected to change in the near future.

Directly embedding Web servers on sensors is a recent development [147, 182]. Examples of embedded Web servers that can be integrated in resource-constrained devices are NanoHTTPD [52] and the Nokia mobile Web server [26].

The important properties of the WoT can be summarized as follows:

- It uses HTTP as an application protocol rather than as a transport protocol as done in the world of WS-* Web Services [13].
- It exposes the synchronous functionality of smart objects through a REST interface and respects the blueprints of resource-oriented architectures (see Section 5.2.3).
- It provides the asynchronous functionality (i.e. events) of smart objects through the use of largely accepted Web syndication standards such as Atom or server-push Web mechanisms such as Comet (see Section 2.6).

Open research questions of the WoT include: local/global discovery of physical devices and services; complete description of these services using standardized languages and semantic techniques; uniform interaction with embedded devices using standard protocols; and asynchronous, Web-based interaction with devices that sense the environment for sporadic events.

2.4 REST and Resource-oriented Architectures

A fundamental part of the WoT is to design services offered by physical devices, using standardized, uniform techniques. In order to do that, WoT engineers suggested the adoption of Web services and more specifically the utilization of REST.

Web services have been extensively used during the last few years to provide integration across heterogeneous, distributed systems. Web services tend to fall into one of two camps: Big Web services (WS-*) and RESTful Web services. WS-* [13] are a set of complex standards and specifications for enterprise application integration. They form the foundational elements for building service-oriented architectures (SOA), introducing new possibilities for large-scale software design and software engineering.

More recently, RESTful Web services [142] have attracted many researchers, engineers and software developers. The notion of REST has been conceptualized in Roy Fielding's PhD thesis [59]. Rather than being a technology or standard, REST is an architectural style which basically explains how to properly use HTTP as an application protocol. REST advocates in providing services directly based on the HTTP protocol itself. An application programming interface (API) fulfilling the REST architectural style is said to be RESTful.

Beyond the basic design criteria provided in Roy Fielding's thesis, the REST community has been working on refining the notions to create resource-oriented architectures (ROA), in order to provide and connect together services on the Web. Because of the underlying simplicity of this architecture, it could be adapted in order to interconnect the physical world and, in particular, embedded devices. Basically a ROA is about four concepts:

- **Resources**. A resource in a ROA is anything important enough to be referenced and linked by other resources. It can be seen as an entity that provides some service.
- **Their Names**. A resource has to be addressable, i.e. it needs to have a uniform resource identifier (URI) that uniquely identifies it. This concept is known as the requirement for addressability of RESTful API.
- **Links between them**. Thanks to the hyperlinking structure of the Web, resources can be connected to one another. This is the requirement for connectedness of RESTful API.
- **Their Representations**. Resources can be represented using various formats. The most common one for resources on the Web is (X)HTML, due to its intrinsic support for browsing and human readability. However, when machine readability is required, XML and JSON are often chosen (see Section 2.5). Note that JSON is a data interchange format which stands as a lightweight alternative to the sometimes too verbose XML.

REST proposes the use of a uniform interface. Resources can only be manipulated by the methods specified in the HTTP standard. Amongst these verbs, four are most commonly used:

- **GET** is used to retrieve a representation of a resource. Concretely, sending a GET request along with the URI *http://.../sensors/officeLamp/Illumination* would return the light level currently observed by *officeLamp*.
- **POST** represents an insert or update of a resource.
- **PUT** is a method to alter the state of a resource. As an example, it could be used to change the state of an actuator. Sending a POST request to *http://.../actuators/bedroomLamp /Light* with the parameter *state=on*, causes the effect of *bedroomLamp* to be turned on.
- **DELETE** is used to delete resources.

Comparing the two trends in Web Services, WoT engineers concluded that RESTful Web services are more appropriate for resource-constrained, ad hoc environments [130, 73]. Thanks to their simplicity, the use of a uniform interface and the wide availability of HTTP libraries and clients, RESTful services are truly loosely coupled. This means that services based on RESTful API can be reused and recombined in a quite straightforward manner.

2.5 Exchange Data Formats on the Web

Exchange of data between two or more applications on the Internet/Web needs to be on a machine-readable form. Many data formats have been proposed, all aiming at the optimization of different aspects (speed, overhead, generality). In the following subsections, we describe the most popular data formats which are employed in machine-to-machine (M2M) communication on the Web. In addition, we explain some basic things about MIME types, which compose a fundamental standard that describes the structure of messages on the Internet.

2.5.1 XML

Extensible Markup language (XML) [23] provides a simple way of how data can be represented in textual form where meta-tags enclose the data item. XML is a markup language that defines a set of rules for encoding documents in a format that is both human- and machine-readable. Figure 2.7 shows an example XML listing two embedded devices: a sensor mote and a smart power outlet. Together with an XSL style-sheet that acts as a grammar describing how to read the XML, almost any kind of data can be encoded.

```
<? xml version = '1.0' encoding = 'utf-8' ?>
<devices>
   <device>
      <name>Living Room Sensor</name>
      <type>Telosb Sensor Mote</type>
   </device>
   <device>
      <name>Television</name>
      <type>Plogg Smart Power Outlet</type>
   </device>
</devices>
```

Figure 2.7: Example XML with a list of physical devices.

2.5.2 JSON

JavaScript Object Notation (JSON) [70] is a data format that claims to be human-readable and at the same time easy to be parsed for machines. It is derived from the JavaScript scripting language for representing simple data structures and associative arrays, called objects.

As opposed to XML, JSON tries to remove some meta-tags, to minimize the amount of characters used for the serialization of an object or data. Although everything in JSON is represented as a JSONObject there are two basic subtypes/structures to distinguish:

- Key/Value pairs. A chunk of data is stored into the JSON data structure together with a lookup key (comparable to a hash-map).
- Lists. A series of data values stored comma-separated within "[" and "]".

Figure 2.8 shows a JSON object (devices) holding a list of two JSON objects (device). Thanks to its simplicity, readability and efficiency JSON has become a vastly accepted data format for the transmission of character data. For almost any programming language there exist JSON libraries to parse and encode JSON objects. Finally, in order to define the structure of JSON data, JSON Schema is a specification for a JSON-based format.

2.5.3 Multipurpose Internet Mail Extensions

Multipurpose Internet Mail Extensions (MIME) [69] is a standard that describes the structure of messages on the Internet. Originally, MIME was invented to embody any kind of data into emails (e.g. images, video, audio), but today it is applied to other areas as well (e.g. HTTP, search meta-data). MIME categorizes format and encoding into types called Internet Media Types or simply MIME types.

The Internet Assigned Numbers Authority (IANA) maintains a list of all the official MIME types[9]. Nonetheless, Web applications are allowed to define their own types. Types contain subtypes to further organize and classify. For example, a MIME type for an image encoded as *png*

[9] http://www.iana.org/assignments/media-types/index.html

```
{
   "devices": [
     {
       "name": "Living Room Sensor",
       "type": "Telosb Sensor Mote",
     },
     {
       "name": "Television",
       "type": "Plogg Smart Power Outlet",
     }
}
```

Figure 2.8: Example JSON object with a list (devices) of two JSON objects (device) holding both two key/value pairs.

is expressed with *image/png*, an audio in *mp3* with *audio/mp3*. Within a MIME document, the MIME type is specified with the keyword *Content-type*.

Documents that use MIME have a header and a body. The body contains the payload (the encoded document(s)) and the header maintains instructions how the payload has to be interpreted. Figure 2.9 shows a sample document encoded as MIME.

2.6 Web Messaging Protocols

The HTTP protocol functions as a request-response protocol in a client-server computing model. It is an example of client *pull* technique, a style of network communication where the initial request for data originates from the client, and then is responded by the server. However, HTTP does not seem suited for applications that involve event-driven scenarios. In such scenarios,

```
MIME-version: 1.0

Content-type: multipart/mixed; boundary ="multipartBoundary"

A MIME message containing two documents. One as XML, one as plain text .

--multipartBoundary

Content-type: text/plain

This document is encoded as plain text .

--multipartBoundary

Content-type: text/xml; charset=UTF-8

<document> This is an XML document.</document>

--multipartBoundary
```

Figure 2.9: MIME encoded document.

push technology is considered more suitable. Server push [60] describes a style of Internet-based communication where the request for a given transaction is initiated by the publisher.

Considering WSN applications, the request/response nature of the HTTP protocols is sometimes inappropriate. Hence, the development of event-driven applications directly over the Web has been enabled by means of Web push tools and techniques. In the following subsections, we present these different push methods for event-driven Web messaging, after we introduce pull-based syndication on the Web.

2.6.1 Web Syndication

Content syndication is a basic form of Web messaging. Feed formats such as as RSS[10], or the more well-defined Atom standard, have become popular formats for exchanging machine-readable data over the Web.

[10]http://www.rssboard.org/rss-specification

The Atom Syndication Format [123] is an XML language used for Web feeds, while the Atom Publishing Protocol (AtomPub) [67] is a simple HTTP-based protocol for creating and updating Web resources. Atom offers a convenient way to deal with time-ordered data, thus is particularly suited for storage, querying and retrieval of stored information. However, Atom suffers from the pulling mode that prevents one to build event-driven applications.

2.6.2 Publish/Subscribe Systems

Publish/subscribe (pub/sub) systems [56] are common and widely used in distributed computing and large enterprise applications. Highly scalable implementations of brokers with a variety of features are available for existing protocols such as JMS[11] or AMQP[12].

XMPP [145] is an XML-based messaging protocol widely used for chat servers. It is based on a decentralized network of servers that provide a multi-hop routing delivering messages from one client to the other. Only recently, sensor networks have began to explore pub/sub messaging to enable complex and interoperable applications that scale. MQ Telemetry Transport (MQTT) is a very lightweight pub/sub messaging transport, ideal for mobile applications because of its small size, low power usage, minimized data packets, and efficient distribution of information to one or many receivers MQTT-S [87] is a messaging protocol designed specifically for WSN.

2.6.3 Comet

An alternative type of Web applications that attempts to eliminate the limitations of the traditional HTTP polling has become increasingly popular. This model, called Comet (also called HTTP streaming or server push), enables a Web server to push data back to the browser without

[11] http://java.sun.com/products/jms/

[12] http://amqp.org

the client requesting it explicitly. A few of these techniques are BOSH[13], Bayeux[14], or reverse HTTP[15], and numerous implementations to support server-side eventing are available.

Since traditional Web browsers were not designed with server-based events in mind, Comet is a hack implemented through several specification loopholes, each with different benefits and drawbacks. One general idea is that a Web server does not terminate a connection after the response data has been served to a client, but leaves it open to send further events immediately to one or multiple clients. Comet servers and clients frequently use named channels or topics which are useful when different objects want to send data to a number of interested parties.

The advantage of Comet is that clients can get real-time notifications from servers with a standard Web browser, even behind firewalls or NAT. The growing need for a push-based communication model is further supported by the introduction of Web Sockets and server-sent events introduced in the upcoming HTML 5 specification.

2.6.4 Web Hooks

Web hooks are another solution for HTTP eventing that enable users to receive events and data in real-time from applications, through user-defined callbacks over HTTP. Once an event occurs, the application will POST data to the callback URLs specified by the users at subscription time. Web hooks enable Web applications to synchronize data with other applications, but also to process, filter or aggregate data from different sources and to notify people. This requires that parties who want to be notified must implement a Web server where notifications will be posted. This mechanism remains a clean solution, unlike the hacks used by Comet, but on the other hand,

[13] http://xmpp.org/extensions/xep-0124.html

[14] http://svn.cometd.org/trunk/bayeux/bayeux.html

[15] http://www.reversehttp.net

Web hooks cannot be used when the client does not possess a network address as is the case when the client is behind NAT or a firewall.

An interesting recent project is RestMS[16], which offers a RESTful interface to the Advanced Message Queuing Protocol (AMQP), and defines the behavior of a set of feed, join, and pipe types that provide an AMQP-interoperable messaging model. PubSubHubbub[17] is a lightweight and open server-to-server publish/subscribe protocol based on Web hooks, as an extension to Atom and RSS. Servers can get near-instant notifications when a topic (feed URL) they are interested in has been updated.

2.7 Summary of the Technologies Adopted

In this section, we describe how the various technologies, listed in this chapter, have been used in order to implement, test or evaluate the Web applications, case studies, installations and deployments that are presented through this thesis.

At first, concepts from the WoT and especially the use of REST and resource-oriented architectures have been harnessed in Chapter 4 to design the general architecture of the application framework for smart homes. XML and JSON are the message exchange formats selected to interact with the physical devices of the smart home.

In Chapter 5, embedded sensor technology is employed to explore and demonstrate the Web-enablement process of home devices. More specifically, sensor motes and smart power outlets

[16] http://www.restms.org

[17] http://pubsubhubbub.googlecode.com

are used to develop a smart home ecosystem for the IoT. These devices form wireless sensor networks inside the home environment, functioning using operating systems dedicated to resource-constrained devices. The physical and media access control layers of these devices operate according to the IEEE 802.15.4 standard while 6LoWPAN is the enabling technology for integrating the IP protocol with these embedded devices.

The WoT and REST are used to model the services offered by these devices in a resource-oriented way, transforming them into tiny Web servers. Except from the Web-based request-response model, sporadic events are pushed from the physical devices to interested third parties (e.g. the application framework), using event-based Web messaging protocols.

Then, Chapter 6 employs sensor motes to analyze the behavior of the request queue mechanism and to identify potential benefits of using request queues in order to manage the interaction with home devices.

Chapter 7 also uses sensor motes in a multi-hop topology to explore the influence of using Web techniques for enhancing the performance of smart home operations, mainly in terms of the response times with the embedded devices and for reducing their energy consumption, preserving their battery. Event handling through Web messaging protocols is also evaluated in this chapter.

Finally, Chapters 8, 9 and 10 present interesting ICT applications for the IoT/WoT, using sensor motes and smart power outlets in proof-of-concept case studies. The communication with social networking sites, smart grid controllers and Web directories for environmental services is achieved through the Web, using RESTful interaction patterns while message exchange is performed using XML and JSON data formats.

Chapter 3

Related Work

From the invention of embedded technology many decades ago until today, numerous researchers and developers envisioned, designed and developed ubiquitous applications, to transform physical environments into smart spaces. Creating digital representations of the physical world is still an open research question, with a plethora of enabling drivers in the latest years, both in hardware and software. Hardware drivers include RFID technology, embedded sensors and actuators, QR codes, smart meters, smart power outlets, even nano-sensors. Software drivers involve middleware and frameworks for supporting interaction with pervasive areas, as well as operating systems and libraries that facilitate communication with smart devices. In this chapter, we focus mainly on software-based research that targets middleware, frameworks and applications for ubiquitous computing and smart home environments. In the following sections, we present considerable work in the field, focusing mostly on research in smart homes. At the end of each section, we explain the contribution of this thesis in relation to the research area in the section under study.

The chapter is organized as follows: Section 3.1 begins with a general overview of ubiquitous middleware and Section 3.2 discusses some early approaches for enabling smart home environments. Section 3.3 identifies pervasive applications that include a request queue mechanism to achieve their goal while Section 3.4 describes efforts that consider smart metering for energy awareness. Then, Section 3.5 mentions projects that aim to achieve energy efficiency in smart homes and buildings and Section 3.6 reviews enabling technologies and solutions for the vision of an Internet of Things. Section 3.7 presents various middleware which are inspired by the Web principles and Section 3.8 refers to large-scale, Web platforms for sensor devices and services. Afterwards, Section 3.9 depicts various developments and prototypes for creating a Web of physical things and, finally, Section 3.10 presents initiatives for the development of smart homes by employing Web technologies.

3.1 Ubiquitous Middleware

Ubiquitous computing (UbiComp) is a post-desktop model of human-computer interaction in which information processing has been thoroughly integrated into everyday objects and activities [175]. Increased context-awareness, provided by high-precision embedded sensor devices, created a culture around ubiquitous applications and services.

Factors such as the high availability of sensor devices and their reduced costs, the extended research performed in algorithms for WSN (covering a broad range of issues such as routing, congestion control, fault tolerance etc.), as well as the existence of open-source, scalable operating systems for sensors and sensor networks led to the development of numerous ubiquitous middleware platforms [33, 53]. These platforms targeted different areas such as industrial control, environmental monitoring, transportation, logistics, health monitoring etc.

One of the most popular and complete ubiquitous platforms is Gaia [144]. Gaia applies the concepts of a traditional operating system to middleware, in order to manage physical devices in a distributed, physical space. It constitutes an experimental middleware infrastructure that supports the development and execution of portable pervasive applications. Aura [152] is an architectural framework, similar to Gaia, but it mainly focuses on supporting mobile users, who are moving around different environments. The philosophy of Aura is that users' tasks are considered first-class entities inside the ubiquitous computing environment. Shaman [148] is a gateway system that enables low-power devices to be part of wider networks. Its novel architecture motivated the implementation efforts of many UbiComp researchers.

Thesis contribution:

The application framework for smart homes (see Chapter 4) is considered a middleware for ubiquitous computing, targeting smart home environments. Its novelty lies on its smart layered design using threads and request queues for handling the communication with home embedded devices. Innovative features of the framework include multi-user support, reliability and fault tolerance, as well as satisfactory performance in terms of response time, because of a fast retransmission mechanism and a caching system. The Web-based architecture of the framework ensures interoperability between heterogeneous smart appliances.

3.2 Smart Homes

Some ubiquitous middleware platforms focused on home environments, transforming residences into smart homes. The majority of them examined the integration of technology and services through home networking for automation and a better quality of life.

Context-awareness in households includes home environmental conditions, such as temperature, humidity and illumination, as well as information about the residents such as their location, mobility tracking and inhabitant behavior.

The Intelligent Home [109] is a simulated intelligent home environment, populated with appliance agents. These agents interact and coordinate to perform home tasks efficiently by sharing resources. The House_n [90] is a living laboratory for the home, with an integrated ubiquitous sensor architecture. The vision of this project was to develop a teaching home, to study technology that motivates behavior change in context.

The Aware Home [101] is another example of a living laboratory for ubiquitous computing research. The technology used includes human position tracking through ultra-sonic sensors, RF technology and video, recognition through floor sensors and vision techniques.

Microsoft's EasyLiving [24] is a middleware for building intelligent home environments based on XML messaging, integrating geometric knowledge of people, devices and places. The adaptive house [120] allows the home to program itself by observing the lifestyle of inhabitants and then learning to predict their needs, by means of neural networks.

The Gator Tech smart house [84] develops and deploys extensible smart house technologies, employing a service-oriented OSGi framework that facilitates service composition. This work incorporates RFID tags, which are attached to electrical devices, to automatically recognize when plugging the devices into outlets and also a smart floor, which serves as a position-only location system. Figure 3.1 is a snapshot of this smart house, where many elements of the house present smart behavior.

The MavHome project [183] employs a hierarchical hidden Markov model for recognizing the activities and behavior of the inhabitants and controlling adaptively the environment, including lights, fans, and mini-blinds. Following a different approach, iDorm [42] also focuses on

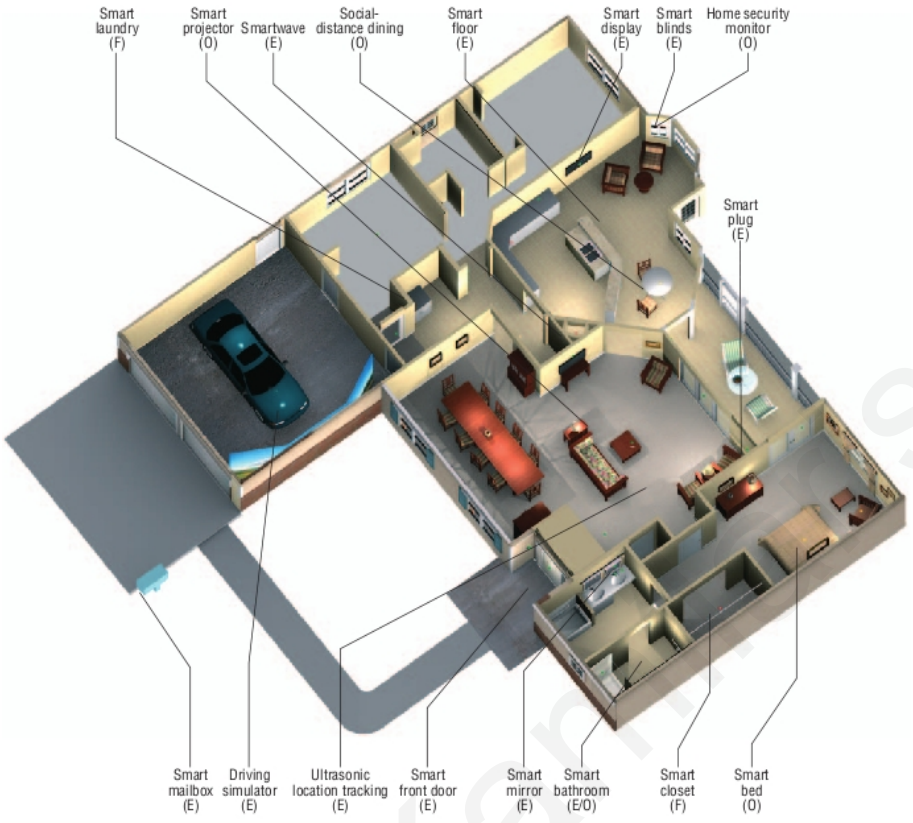

Figure 3.1: The Gator Tech smart house.

automating a living environment, however, instead of a Markov model, it models inhabitant behavior by learning fuzzy rules that map sensor state to actuator readings representing inhabitant actions. The setting is a campus dorm environment, which is automated using fuzzy rules learned through observation of inhabitant behavior.

Finally, the work performed in [155] demonstrates that ubiquitous, simple sensor devices can be used to recognize activities of daily living from real homes. It is an early attempt to recognize activities of interest such as toileting and bathing, by means of naive Bayes classifiers, with detection accuracies ranging from 25% to 89%, depending on the evaluation criteria used.

Thesis contribution:

While the presented approaches focus on home automation as well as activity and behavior recognition, our application framework for smart homes (see Chapter 4) focuses on addressing the heterogeneity of home embedded devices in terms of hardware and software, ensuring reliability, fault tolerance, acceptable performance and support for multiple residents who may interact simultaneously with their smart home.

3.3 Using Request Queues in Pervasive Computing

Various pervasive applications employed queues for handling the management of requests and for providing Quality of Service (QoS) guarantees regarding service execution. A popular platform for messaging handling is Apache ActiveMQ[1], which is a messaging broker platform that employs message queues for supporting public/subscribe architectures, load balancing and fault tolerance etc.

A middleware using FIFO M/M/m queues to offer dynamic adaptation to applications including multiple users and servers was presented in [27]. A certain level of QoS may be maintained by adding redundant servers and by performing bandwidth balancing between the servers. The work in [173] considers a delay-tolerant mobile sensor network (DFT-MSN) for pervasive information gathering. A DFT-MSN is an occasionally connected network that may suffer from frequent partitions. In this work, each sensor has a data queue that contains data messages ready for transmission when an opportunity appears. The proposed queue management scheme takes into account fault tolerance and signifies how important the messages are, to develop simple and efficient data delivery schemes.

[1] http://activemq.apache.org/

Concurrent event detection for consistency checking of pervasive context in asynchronous environments is achieved in [85], by using message queues for modeling the pervasive application as a loosely-coupled message-passing system. Shaman [148] is a gateway for sensor devices that uses request queues to handle incoming/outgoing messages from/to the sensor devices. The architecture of Shaman influenced the design of the application framework for smart homes, which is proposed in this thesis.

A scheduling algorithm to complete executing tasks within predetermined time deadlines and constraints, when multiple processors (e.g. Web servers) are available to execute these tasks was described in [88]. The scheduler is modeled as a global request queue that forwards requests to the local waiting queue of each processor. Similarly, a middleware service addressing the timely provisioning of mobile services in critical pervasive environments was proposed in [128], considering client/server lifetimes, server current load and periodic service requests. Specifically, the scheduler service of each mobile server manages FIFO request queues, which are scheduled according to the classical round robin algorithm.

Finally, the work in [181] targets home environments, by modeling all the tasks of the home network as a combination of handling events. Priority management is achieved by assigning priorities to the events while an event kernel handles priorities using multi-level priority queues, prioritized event dispatching threads and an event scheduler.

Thesis contribution:

Our request queue mechanism (see Chapter 6) differs from related work by directly associating request queues with embedded devices that are deployed inside smart homes. This queuing mechanism provides reliability and offers numerous potential benefits to pervasive applications that

operate inside the home environment, such as priority-based request execution and load balancing. Shaman [148] presents a similar concept, however, the use of the queues is only rudimentary, without any analysis of the queuing behavior. On the contrary, our work involves a detailed analysis of the request queue mechanism and the setting of design parameters (e.g. the request queue retransmission interval).

3.4 Smart Metering for Energy Awareness

As already mentioned in Section 2.1.2, there exist two prevalent approaches for household energy monitoring: whole-home by means of residential smart meters; and device-specific monitoring by using smart power outlets.

Whole-home approaches are used to measure the total energy consumption of the house. In [79, 114], circuit-level power measurements are used to separate aggregated data into device-level estimates, with accuracy more than 90%. ViridiScope [102] places inexpensive sensors near electrical appliances to estimate their power consumption. Through experiments in a real house, ViridiScope shows that it estimates end-point power consumption within 10% error.

The study in [154] combines office and domestic energy usage and aggregates this usage to estimate the electricity footprint of people. It is one of the first studies that attempts to create electrical consumption profiles of citizens, analyzing their energy demands, both at home and at work. According to the study, 55% savings in an office is possible, from just a more rational duty cycling scheme of the desktop machines and the display units.

Google PowerMeter [66] uses energy information provided by utility smart meters and energy monitoring devices, to assist users in viewing their home's energy consumption online, in interactive graphs. Google recently announced that it has discontinued this project. Microsoft Hohm [117] gives online recommendations to residents for home energy efficiency.

The eMeter [116] employs a smart electricity meter to provide real-time feedback of the total consumption of a house on a mobile phone. The system can disaggregate overall electricity consumption by forcing the users to manually switch specific devices on or off, in synchronization with the mobile application. In Figure 3.2, a user is measuring in real-time the power consumption of his electrical appliances, through his mobile phone.

Figure 3.2: The eMeter mobile application [116].

Device-specific techniques plug smart power outlets in individual electrical appliances. ACme [93] is a high-fidelity AC metering network that uses wireless sensor nodes equipped with digital energy meters to provide accurate energy measurements of single devices. EnergieVisible [43] employs smart meters to visualize in real-time the energy consumption of the appliances that are connected to the meters, in a Web-based interface, converting the process of monitoring and comparing electrical consumption of home appliances into an easy task (see Section 4.3.3).

EnergyLife [91] utilizes a mobile application to provide electricity consumption feedback and conservation tips. The Energy Aware Smart Home [92] allows users to use their mobile phones as magic lenses to view and analyze the energy consumption of their appliances just by pointing on them with the phone's camera.

PowerPedia [176] envisions a collaborative energy encyclopedia. PowerPedia encourages users to compare the consumption of their electrical appliances to those of others. Thus, it helps

users to better estimate the consumption of their appliances and take effective countermeasures to save energy. WattBot [132] performs analytical energy monitoring by utilizing multiple clamp meters. Each clamp meter monitors a separate circuit of the home electrical network.

Smart metering includes not only electricity monitoring but also monitoring of other resources such as water and gas. For example, HydroSense [61] is a low-cost, easily-installed, single-point sensor of pressure within a home's water infrastructure. It supports identification of activities inside the home that involve water and also estimation of the amount of water being used at each activity with 97.9% aggregate accuracy. Figure 3.3 depicts this pressure sensor. GasSense [32] analyzes the acoustic response of a home's gas regulator, being able to sense both the individual appliance at which gas is currently being consumed as well as an estimate of the amount of gas flow. Initial results show that GasSense has an average accuracy of 95.2% in identifying individual appliance usage.

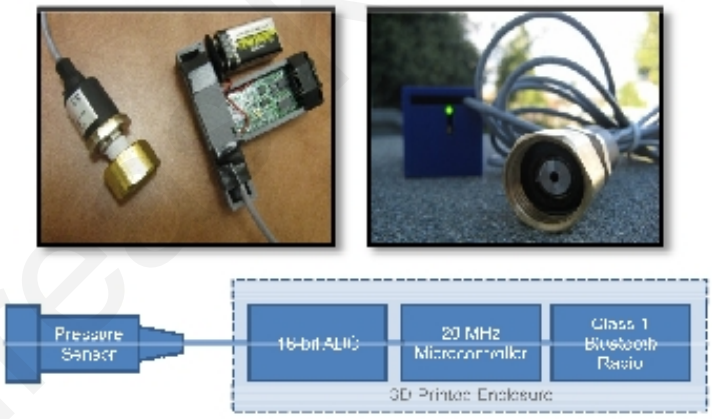

Figure 3.3: The HydroSense prototype pressure sensor implementation [61].

Thesis contribution:

Our application framework for smart homes (see Section 4.2) integrates Ploggs smart power outlets (see Section 5.1.2) and enables them to the Web (see Section 5.2), in order to monitor the electrical consumption of individual electrical appliances and control their operation by switching them on/off, through a Web API and also by means of a Web graphical interface (see Section 4.3.2). Moreover, the functionality of the electrical appliances of the smart home may be programmed and automated by using physical mashups (see Section 4.2.6) or through a graphical mashups editor (see Section 4.3.4). Smart metering is also employed in a social competition for energy conservation in neighboring flats (see Section 8.2).

Furthermore, the vision of PowerPedia [176] may be parallelized with the concept of Social Electricity Facebook application (see Section 8.3). However, while PowerPedia focuses on sharing the energy profile of various electrical appliances, Social Electricity aims at sharing and comparing the energy consumption of people in a social networking application.

3.5 Energy-Efficiency in Smart Homes

Energy-efficient techniques for smart homes and buildings basically combine smart metering with local environmental context to gain advanced automation and save energy.

iPower [162] adjusts devices to automatically reduce unnecessary energy consumption by means of WSN and power-line control devices. Sensors are deployed in each room to monitor the usage of electric appliances and to help the control server to determine if there are electric appliances that can be turned off to reduce unnecessary energy consumption. The general architecture of the X10-based system is shown in Figure 3.4.

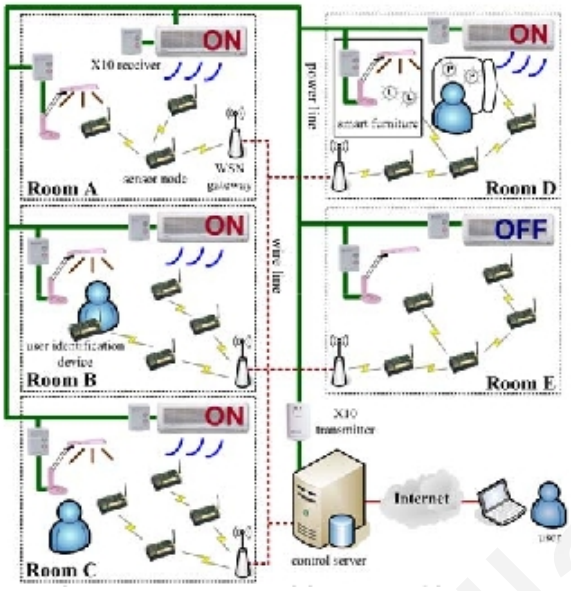

Figure 3.4: The system architecture of iPower [162].

Following a similar direction, the work in [18] creates profiles of the behaviors of house residents through presence detection by means of infrared sensors. Then, it automatically optimizes energy consumption and cost by means of a prediction algorithm, while guaranteeing the required comfort level. When users change their habits due to unpredictable events, the system is able to detect wrong predictions by analyzing in real time information from sensors and modifying system behavior accordingly.

The authors in [55] deploy a wireless camera sensor network to obtain occupancy and usage patterns in a building, in order to intelligently control the lighting and HVAC system and save energy. It is stated that their system estimates occupancy with an accuracy of 80%. Occupancy estimates and usage patterns allow a 14% reduction in HVAC energy usage.

Context-aware power management (CAPM) is explored in [83]. User location is harnessed to effectively manage energy-consuming devices in a building. The objective of CAPM is to minimize the overall energy consumption while maintaining good user-perceived device performance.

Experiments show that it is possible to get 6% savings, but these savings are highly dependent on user behavior. By adding more sensors to improve context inference, the overall energy consumption can actually increase.

A thermostat that is aware about the current location of the user through the global positioning system (GPS) sensor of his mobile phone is proposed in [78], and may lead to savings of 7% for households that do not regularly use the temperature setback afforded by manual and programmable thermostats. The authors discuss that these savings may be obtained without requiring any change in occupant behavior or comfort level, and the technology can be implemented affordably by exploiting the ubiquity of mobile phones.

The work in [171] examines the potential of user-created policies and their application to develop a system that supports the user in gaining awareness about energy consumption habits and saving potentials, without compromising his standard of living. An interesting aspect of this work is the categorization of system-level policies as tariff-dependent policies (taking into account the information on pricing schemes published by utilities), device-dependent policies (permanent, stand-by, ad-hoc devices), and threshold-dependent policies (policies based on limits put on certain environmental conditions).

Finally, two high-profile sustainable home deployments are described in [113]. The goals of these deployments are to help residents to gain awareness of resource use, facilitate efficient control of house systems and to encourage conservation in daily activities. Their criteria in designing these smart homes are: real-time feedback; contextually appropriate information presentation; individual and social motivation; control; aesthetics; and familiarity. Figure 3.5 is a snapshot of the user interface, developed to inform the residents in detail about their resource consumption, giving them incentives to save electricity and water.

Figure 3.5: Resource-aware user interface of a sustainable home deployment [113].

Thesis contribution:

Even though this thesis does not claim to offer an effective approach for energy conservation in smart homes, various applications and case studies are presented through the thesis, which may help people to conserve electricity, mainly by means of energy awareness. Examples include a graphical Web interface for real-time feedback of the consumption of individual electrical appliances (see Section 4.3.3) and a Web application for developing smart rules for energy conservation (see Section 4.3.4). In addition, encouraging findings for energy saving are obtained from a social competition in blocks of flats (see Section 8.2), while Social Electricity, as a Facebook application for social comparisons of energy consumption, provides interesting initial feedback from consumers of electricity (see Section 8.3). Finally, potential savings may be achieved by means of a task scheduler that exploits the demand response program of the smart grid for programming energy-related tasks in low-tariff hours of the day (see Section 9.2).

3.6 Enabling the Internet of Things

Many projects are inspired from the notion of the IoT and try to interconnect the physical world with the Internet. The work in [174] augments physical objects with RFID tags, to create virtual representations of these objects. The project JXTA [158] specifies a standard set of protocols for ad hoc, pervasive, peer-to-peer computing. It can run on a wide range of devices including servers, PCs, and memory-constrained embedded devices. JXTA is among the first real attempts to Internet-enable physical objects.

Social devices [169] are special kinds of augmented objects that use the Internet in order to promote socialization, look smarter and better serve users. A typical example, shown in the left side of Figure 3.6, is a prototype umbrella which is able to obtain current weather conditions through communication with surrounding rain sensors, being also able to download the weather forecast from the Internet, in order to provide better advice to the user. The umbrella reacts when the user leaves home without taking it, by issuing a synthesized voice alert.

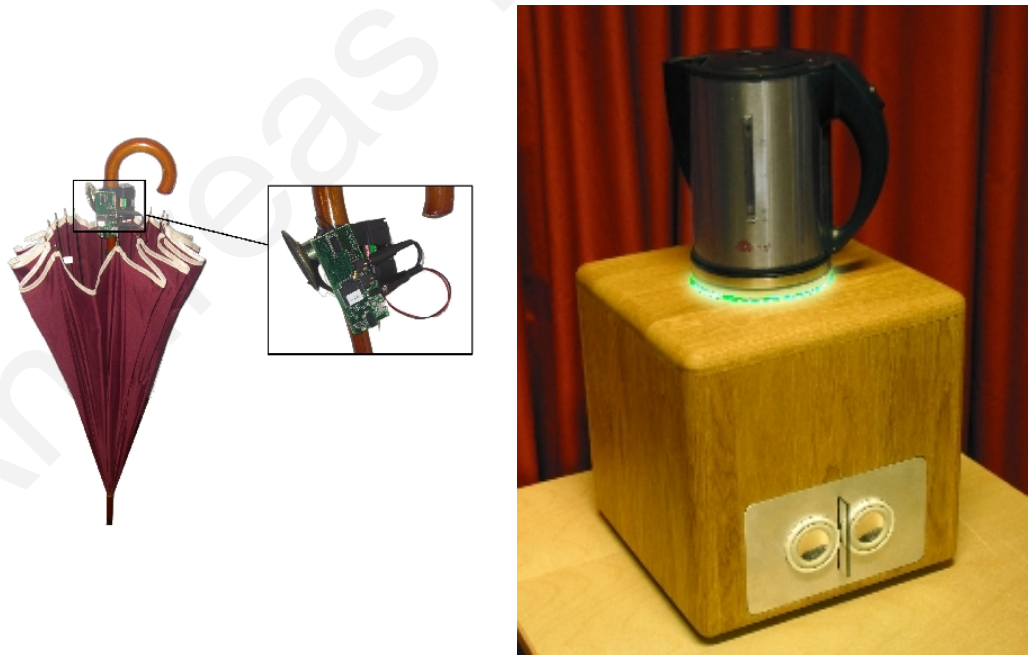

Figure 3.6: The aware-umbrella prototype [169] (left) and a tea kettle prototype [94] (right).

Internet-connected physical smart objects are also presented in [168]. As an example, a *real-widget* embodies the capabilities of computer-based widgets into the real world. The *iCompass* is a device that acts like a compass, but instead of pointing North, it points to a zone of a background dial card that represents some concrete Internet information. iCompass and example weather dial cards are depicted in Figure 3.7.

Figure 3.7: The iCompass and example weather dial cards [168].

Flexeo [166] is a flexible architecture for connecting wireless sensor networks to the Internet, by distributing the intelligence and the decision making process in different layers. The Flexeo architecture is demonstrated in some prototypes, for industrial and user activity monitoring scenarios. The work in [94] describes how home appliances might be enhanced to improve user awareness of energy usage. The authors believe that electricity usage reduction is a challenge that is best met through the design of interfaces that allow users to better control their appliances and inform them of their actions. A fully functional prototype tea kettle with a socially-aware interface is presented, which exploits its rotating base to inform indirectly home occupants about electricity load conditions from a number of interconnected households to achieve electricity load balancing. We can observe this kettle in the right side of Figure 3.6.

Tales of Things[2] is a tagging service that uses QR codes and RFID tags to enable people to attach stories and memories to physical objects. The scanning of readable and writable tags allows stories to be replayed and added. My2Cents [98] is a community-based mobile application that supports online consumer opinions and feedback about retail products, by combining the concept of microblogging with mobile product recognition. Finally, Fosstrak[3] is an open-source RFID software platform for discovering and managing RFID-based information.

Thesis contribution:

This thesis contributes to the IoT research through the implementation of an IPv6/6LoWPAN wireless network of sensor devices and its deployment inside a proof-of-concept smart home (see Chapters 6 and 7). Our evaluation efforts indicate that an Internet-based strategy, using the IPv6 protocol, offers acceptable performance in smart homes involving IPv6-enabled sensor devices. Applications concerning social networking of the smart home (see Chapter 8), integration of smart homes to the smart grid (see Chapter 9), global discovery of environmental services through DNS (see Section 10.3) as well as knowledge inference in the urban environment (see Section 10.4), constitute interesting demonstrations of the IoT (and respectively the WoT) in real-life scenarios.

3.7 Web-Based Ubiquitous Middleware

The following middleware applications differ from the ubiquitous middleware platforms listed in Section 3.1, as they have been designed and implemented using Web technologies,

TinyREST [112] is a gateway that follows the REST principles, to enable sensor networks to the Web. Its main drawback is that it violates REST by introducing the extra verb *SUBSCRIBE*.

[2] http://www.talesofthings.com/

[3] https://code.google.com/p/fosstrak/

pREST [45] is a similar effort, which defines a RESTful protocol to bring the simplicity and holistic view of data and services on the Web to pervasive systems. Following this direction, the work in [153] focuses on the scalable discovery of heterogeneous sensor devices on the Web, augmenting the proposed discovery mechanism with semantic Web elements.

Prehofer et al. [135, 164] describe a Web-based middleware for smart spaces, which strongly relies on technologies used in Internet services. In the local smart space, the Zeroconf mDNS mechanism is used to support service registration and discovery. In [136], the previous Web-based middleware is extended with a flexible access control mechanism on top of OpenID and OAuth, a search engine that can collaborate with existing service and network discovery mechanisms, as well as delivery context client interfaces (DCCI), which constitute an emerging W3C standard that can be used to share context information locally. Its general concept of an IP-based smart space is displayed in Figure 3.8.

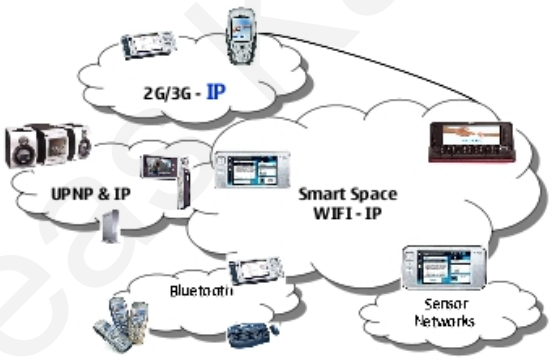

Figure 3.8: Smart Space: Local IP network with smart devices acting as proxies [135, 164, 136].

Thesis contribution:

Our application framework for smart homes (see Section 4.2) constitutes another middleware that operates using Web technologies, following REST principles. It differs from related work

by adopting a layered design, using request queues as a mechanism to handle the interaction with the smart home ecosystem. Furthermore, it offers some additional features such as a Web graphical interface (see Section 4.3) and an open API that promotes the development of physical mashups (see Section 4.2.6). Techniques such as fast retransmissions, caching, load balancing and support for prioritized requests enhance the performance of the framework and ensure reliability in the ubiquitous environment of smart homes. Finally, event-based messaging in a Web-based ubiquitous environment is supported by the application framework (see Section 5.2.4) and it is an issue not considered in related work.

3.8 Web-based Large-Scale Sensor Platforms

Online, Web-based, global sensor infrastructures have appeared in the last decade, to accommodate the large volumes of sensory data that are being produced by Web-enabled sensors and actuators worldwide. These infrastructures act as data brokerage platforms, managing millions of data points per day from thousands of individuals and companies around the world. They enable people to share, discover and monitor in real-time sensor, energy and environmental data from objects, sensors and buildings that are connected to the Internet, around the world.

These platforms promote the Web-enablement of sensors and allow the propagation of sensory readings to interested third parties. Sensors are presented inside satellite maps, in the absolute position where they are located. These sensors can be fixed or mobile.

The most well-known, active platforms available today are Cosm [82], Paraimpu[4], ThingSpeak [4], Sen.se [3] and SenseWeb [146]. A snapshot of Cosm can be seen in Figure 3.9. In this snapshot, a large number of sensors can be observed in the absolute position where they are located

[4] http://paraimpu.crs4.it/

around the world. A key feature of some of these online directories is that they provide open Web API that urge the development of smart applications.

Figure 3.9: A snapshot of Cosm [82].

A drawback of these infrastructures is their centralized nature, with a single point of failure. Decentralized approaches have been also proposed such as IrisNet [63], which uses a hierarchical architecture for a worldwide sensor Web. G-Sense [131] is a peer-to-peer system for global sensing and monitoring. These approaches, although more robust and scalable, have not been largely adopted yet by the public. Moreover, sensor discovery is still a challenging issue.

Other platforms target specific types of embedded devices. For example, Tales of Things[5] is a Web platform where people link their physical objects via small printable tags known as QR codes. The EPCglobal Network[6] is a community of trading partners engaged in capturing, sharing, and discovering Electronic Product Codes (EPC) of products, stored on RFID tags.

[5] http://talesofthings.com/

[6] http://www.gs1.org/epcglobal

Thesis contribution:

Social Electricity Facebook application (see Section 8.3) can be considered a Web-based large-scale platform for sensory measurements related to electricity consumption. This data, acquired by thousands electrical meters in Cyprus, is shared between Cypriot citizens to perceive their electricity footprint. Electricity-related data is currently provided off-line every two months by the Electricity Authority of Cyprus, but in the future it could be provided by smart meters in near real-time. Social Electricity maintains a social character, allowing people to compare their electrical consumption through online social networking.

3.9 Enabling the Web of Things

In this section, related efforts for creating and developing a WoT are listed. One of the first projects envisioning Web presence for physical elements was the Cooltown project [103]. In this project, each thing, place and person has an associated Web page with relevant information.

Priyantha et al. [137] use WS-* [13] to enable a WSN. Undoubtedly, WS-* are not very energy-efficient for operation in constrained environments, since they employ heavy SOAP/XML payloads in messaging. The work in [68] quantifies this statement in terms of time and energy. A recent study [73] compares REST with WS-*, to conclude that REST is more suitable for embedded environments

A path towards the integration of things into the Web is discussed in [178], where physical objects are made available through RESTful principles. As pointed out, no need for any additional API or descriptions of resource/function would be necessary when embedded devices are Web-enabled *by design*. This is one of the first projects envisioning the WoT.

Following these guidelines, Yazar et al. [182] implement an IP-based sensor network system where nodes communicate their information using RESTful Web services. Similarly, Schor et al. [147] develop a 6LoWPAN-enabled WSN, directly integrated into the IPv6-based future Internet. Sensor nodes offer RESTful APIs while service discovery is based on Apple's mDNS proposal. Figure 3.10 presents the general architecture for the integration of 6LoWPAN-enabled sensors with the IPv6 backbone and with existing non IP-enabled devices.

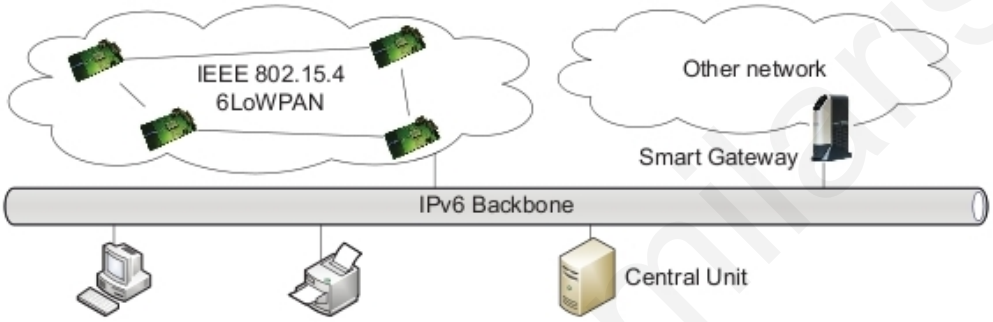

Figure 3.10: Architecture of a 6LoWPAN-enabled WSN [147].

Guinard et al. [75] introduce the concept of *physical mashups*, which builds on the success of Web 2.0 mashup applications to develop a similar approach for integrating real-world devices to the Web, allowing them to be easily combined with other virtual and physical Web resources. The authors demonstrate this concept by using sensors capable of monitoring and controlling the energy consumption of household appliances, offering a RESTful API to their functionality. A nice prototype application is an ambient meter, which changes its color according to the energy consumption at the place where it is located.

A framework for the development of physical mashups through mobile phones is proposed in [71]. The proposed framework is shown in Figure 3.11 and is composed of four main parts: Web-enabled devices tagged with small 2D bar-codes, to be identifiable from mobile phones; virtual

services on the Web; the mashup server framework; and the mobile mashup editors, which allow users to create their mashups in an easy manner.

Figure 3.11: A physical mashups mobile framework [71].

Trifa et al. [160] build a location-aware infrastructure for embedded devices using Web-enabled gateways. By linking gateways together, one can create a hierarchical structure of physical locations that can be used to search for mobile devices based on their location and dynamic information, and not only on keywords. WebPlug [127] is a framework for the WoT that proposes new resource-oriented concepts for interacting with real-world objects, such as typed resources, collections of sensory values, meta-URLs for addressing past versions of a resource, history of sensor readings, observable resources, arithmetic expressions, evaluators and pollers for push-functionality for non-observable resources.

Web approaches for generating and accessing product memories are discussed in [149]. A product memory is an RFID-based mechanism that provides a digital diary of the complete product life cycle of individual products. Sharing of physical things between people that know and trust each other is proposed in [44]. Trust is delegated from the owner of some physical devices to his online contacts, which are discovered from his social networking structures, by exploiting the open, Web API of social networking sites.

Dyser [125] and Snoogle [172] are search engines that enable finding real-world entities such as sensor devices in real-time. Web-based publish/subscribe messaging for sensor infrastructures, based on *push* techniques modeled through REST, is proposed in [161].

SOAM [167] is a model for the creation of pervasive smart objects that use semantic Web technologies. These smart objects manage three different types of information structures: context information; capabilities; and constraints. XML schemas, RDF and OWL ontologies, and adaptation profiles are employed to exchange, represent and process information. SOAM is an effort to create a model for automatic environment adaptation to user preferences.

The work in [74] integrates the global EPC Information Services Standard (EPCIS) network into the Web by designing a RESTful architecture for the EPCIS. Using this approach, each query, tagged object, location or RFID reader gets a unique URL that can be linked, exchanged in emails, browsed, bookmarked etc.

Finally, guidelines for architecting a mashable WoT are provided in [76]. The authors discuss several prototypes, implemented using Web principles to connect physical devices to Web. In Figure 3.12, a general framework is provided for the Web integration of embedded devices.

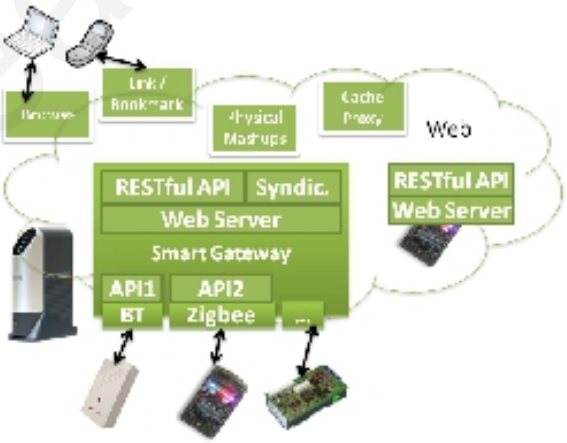

Figure 3.12: Web integration of embedded devices using gateways (left) or directly (right) [76].

Thesis contribution:

This thesis is closely related to the concepts of the WoT. Our application framework for smart homes (see Section 4.2) is an application of the WoT in a smart home environment, enhanced with advanced features such as a request queue mechanism, for increasing the reliability and performance of WoT-related smart home operations. The procedure of Web-enabling home devices (see Chapter 5) addresses important open research issues, such as local device discovery and service description. The concept of physical mashups, proposed in [75] is extended and adapted in this thesis for smart homes. This is achieved by exposing the functionality of the smart home and its home devices as an open Web API and also through the development of a Web-based graphical mashups editor (see Section 4.3.4). Physical mashups are further extended into urban mashups, to comply with the dynamics of urban environments (see Section 4.2.7). Furthermore, our work discusses the practice of sharing Web-enabled home devices through online social networking (see Section 8.1) This concept is similar to the work in [44]. However, that work targets mainly the authentication and access control of Web users, who may share and interact with physical devices, while our application penetrates in social networking sites, to propose a shared, social experience between members of a smart home.

Applications that exploit social networking infrastructures for device sharing and energy awareness (see Chapter 8), as well as applications dealing with the integration of smart homes to the smart grid (see Chapter 9), global discovery of environmental services through DNS (see Section 10.3) and knowledge inference in the urban environment (see Section 10.4), can be considered demonstrations of the applicability of the WoT in the real world.

Finally, important contribution in this thesis is the experimental evidence that by employing Web techniques in IPv6-based smart home deployments (e.g. Web caching, event-based Web

messaging), satisfactory performance may be achieved (see Chapter 7), which is equivalent or even better than native (and more optimized) techniques (such as simulated REST).

3.10 Smart Homes and the Web

The increasing acceptance of the WoT and the potential created from the practice of Web-enabling physical devices and everyday objects, inspired researchers to apply the Web principles in the smart home domain. At first, smart home projects incorporated WS-* for interactions with household devices. In [10], an infrastructure for domestic networks based on WS-* is proposed, to address device heterogeneity. The author considers the role of Web services for interoperability of home appliances, after listing different scenarios for home automation, such as the X10 protocol and closed gateways from remote service providers. The concept of this approach is displayed in Figure 3.13.

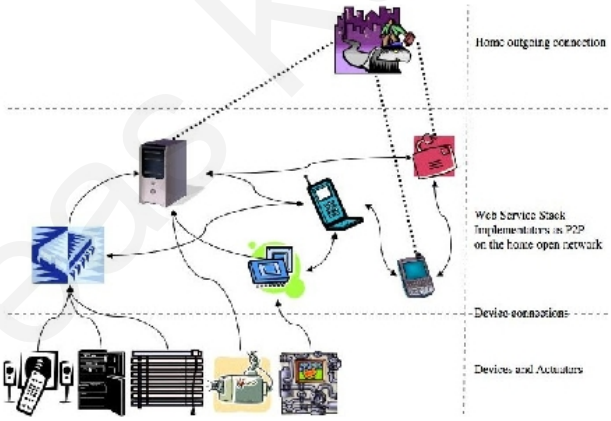

Figure 3.13: WS-* based home architecture in [10].

In a similar study, a service-oriented architecture for smart home environments, based on Open Services Gateway Initiative (OSGi) and mobile agent technology is proposed in [180]. The main architecture is a peer-to-peer model based on multiple OSGi platforms, in which service-oriented

mechanisms are used for system components to interact. Mobile agent technology is applied to augment this interaction and is mainly intended for dynamic situations.

The survey in [65] presents the current emerging architectures and technologies, suitable for wireless home area networks. ZigBee, Z-Wave, INSTEON, Wavenis, and IP-based solutions (6LoWPAN) are thoroughly investigated and compared, in terms of physical characteristics, communication modes, networking, security etc. As stated in the survey *"the increasing functionalities of some solutions and convergence toward IP suggest that future smart home applications will benefit from enhanced quality, security, and interoperability"*.

In another related survey [105], IPv6 and 6LoWPAN are preferred over a number of existing technologies for smart homes (X10, KNX, ZigBee and digitalSTROM). This comparison is shown in Table 3.1. As the authors note, IPv6-based WSN is mature and future-proof, with low cost, easy installation using wireless embedded devices, auto-configurable, offers wide-scale connectivity, natural user interaction and security. The authors claim that Internet technology, utilizing IPv6, can become the future standard in home automation.

Table 3.1: Home automation solutions in comparison to IPv6 [105].

	X10	**KNX**	**ZigBee**	**dS**	**IPv6 (6LoWPAN)**
Network size	2^8	2^{16}	2^{16}	2^{16}	2^{64} per subnet
Data rate	20b/s	9.6kb/s	20..250kb/s	200b/s	250kb/s..1Gb/s
Interface	custom solutions	App. gateway	App. gateway	Web services	UDP, TCP, RESTful Web
Maturity	1975	2002 (1990)	2004	2010	1998 (1969)
Costs	low	high	medium	medium	low
Installation overhead	low	high	low	medium	low
Connectivity	low	medium	medium	medium	high
Security	none	high (EIBsec)	medium (AES)	low	medium (6LoWPAN AES only)

Thesis contribution:

A main contribution of this thesis is the development of an application framework for smart homes (see Section 4.2), developed using Web technologies, following the principles of REST and resource-oriented architectures. It is one of the first -to our knowledge- ubiquitous middleware for smart homes, which employs Web techniques *by design* to achieve interoperability among heterogeneous home devices and smart home solutions. A similar concept was proposed in [10], however, the author suggests WS-* for interacting with information appliances. Their work is more theoretic, without an evaluation. Nonetheless, there is strong evidence to let us believe that REST is more suitable for such an initiative [73, 68]. We demonstrate that by exposing the smart home as an open Web API, physical mashups can be developed for automating the home environment (see Section 4.2.6), promoting application development by end users with increased flexibility and ease of use. Finally, by installing a Web cache on the framework, the performance of the smart home operations is enhanced (see Chapter 7).

Chapter 4

Building a Web-based Application Framework for Smart Homes

This chapter presents the general architecture and methodology for creating Web-based smart home middleware and applications. At first, Section 4.1 identifies some general requirements for building smart home applications. Then, Section 4.2 explains the design and implementation of a Web-oriented application framework for smart homes and defines physical and urban mashups, demonstrating through examples how simple and elegant smart application development becomes, by employing the mashup paradigm. Finally, Section 4.3 discusses the extension of the application framework into a flexible, Web-based smart home application with a graphical interface and presents two graphical applications, one for detailed energy monitoring through real-time feedback from electrical appliances and another for creating smart rules, using the physical mashup example, in order to automate the home environment towards energy conservation.

4.1 Requirements for Future Smart Homes

We believe that future generations of embedded devices will be much more ubiquitous, highly integrated in our everyday lives, massively deployed in home environments. This would create the need for a greater degree of flexibility in order to be manageable. We envision three distinct

interaction possibilities with home devices that cover the most important use cases that should be supported in future smart homes:

- **Ad hoc interaction**. Users directly access sensors and actuators on demand (in a request-response model).
- **Continuous monitoring**. Devices stream data at regular intervals (e.g. smart meters that measure energy consumption of electrical appliances).
- **Event-based messaging**. Events are sent sporadically when something important happens (e.g. indication of fire). Devices must send data only if the phenomenon occurs and this as quickly as possible.

Future smart homes shall be open and accessible for simultaneous users (family members), who could pull easily the data they need directly through the Web, and use them right away in their own applications. We propose a list of requirements that most of the existing smart home application frameworks for embedded devices lack, and will be needed in large-scale, open, and heterogeneous smart home environments. We divide these requirements in three distinct categories: requirements for home devices; requirements concerning the design and development of the application framework; and requirements about performance. Requirements concerning home devices will be considered in Chapter 5, during the process of enabling them to the Web.

Device-Specific Requirements:

- **Multi-hop wireless communication** between home devices, forming a wireless sensor and actuator network in the smart home.
- **Plug and play functionality** of home devices. Devices should operate seamlessly in the home environment, as soon as they are turned on, whether they are stationary or mobile,

indoor or outdoor. There must be standardized methods and techniques for discovering them locally, describing their services, communicating with them, even sharing them between family members, friends or colleagues.

- **Ad hoc interaction** with home devices. The framework needs to interact with devices and their services with minimal prior knowledge about them, without requiring to install drivers or specific applications.

- **Uniform access** to heterogeneous embedded devices. Devices expose their services in well-understood, easily accessible, clean and open API. This enables their smooth integration in smart home applications. The ubiquitous space becomes a cloud where any device can be individually accessed in standardized ways.

Application Framework Design Requirements:

- **Particularities of resource-constrained environments get abstracted**, offering limited reliability (not always reachable devices, variable QoS, device mobility, device failure).

- **Masking transmission failures** to/from the embedded devices, supporting fast retransmissions. This procedure should be invisible to the Web client.

- **Data from embedded devices is open and easily exported** into third-party applications in standard formats (e.g. XML, JSON). All devices and applications shall speak the same language.

- **Interoperable programming interface** that provides the primitives to end users with little programming experience to perform advanced tasks. This would allow developers, tech-savvy users, researchers, even end users to employ any programming language they are familiar with, to develop their own smart rules and automate their home environment.

- **Graphical user interfaces** that allow the management and control of home devices, as well as interaction with their services easily and fast, by home residents who may have no programming experience. Composition of pervasive services can be promoted by graphical editors that allow the creation of smart rules for home automation. Schedulers can program tasks for future execution and notifications can be triggered in cases when important events are sensed.

- **Real-time feedback**. Concerning electrical appliances in particular, the smart home needs to inform residents in real-time about their state and energy consumption, permitting users to switch them on/off to save energy. Electricity consumption data should be available in standard formats.

- **Layered architecture and flexible design** based on well-known methods and technologies. Such design would facilitate, promote and accelerate the development of smart applications, tailored to solve specific challenges in home environments, such as energy awareness and conservation and environmental awareness.

- **Lightweight implementation**. The framework must have a small memory footprint, to be easily installed on an existing computing machine of the smart home e.g. a computer, tablet, router, embedded PC, even a smart meter. Also the protocols which will be running on the embedded devices of the home environment need to be as light as possible.

- **Direct access for residents** to their home environment. Users should be able to access and use their home devices without the need for installing additional software. This would ensure the usability of the framework and it would minimize the entry barriers for users.

- **Concurrent, multi-resident support** through the Web. Residents should be able to interact simultaneously with their household appliances, without loss of requests due to concurrency in the embedded space of home devices.

- **Support for prioritized requests** from home residents or from the application framework. Requests marked as critical need to be satisfied as fast as possible, because their execution is urgent. These urgent requests can include turning off of a faulty appliance to avoid electricity leakage or even fire.

- **Support for streaming events** to interested third parties. The framework must be capable of monitoring the home environment for events and informing users, who may be interested in these events, efficiently and fast. The streaming mechanism must be scalable, allowing multiple events to be sensed simultaneously, but also it should support multiple home residents, acting as subscribers, who may want to be informed urgently about these events.

Performance Requirements:

- **Small waiting times for concurrent requests** from various home residents that target the same home devices. Techniques such as load balancing could be employed for reducing waiting times for residents.

- **Reliability** in regard to request satisfaction. Requests need to eventually be satisfied, even though previous requests may be served and/or transmission failures occur in the wireless medium.

- **Acceptable performance in terms of response times** from embedded devices. Especially since home devices might be multiple hops away from the framework and they can incorporate the TCP/IP stack (for interoperability reasons), smart techniques must be employed for achieving satisfactory performance, comparable to traditional smart home technologies. Response times must be less than a second, even when numerous home residents interact with their home environment. Even in scenarios including high percentages of transmission failures, fast retransmission mechanisms could be considered for masking these failures.

- **Satisfactory battery lifetime** of embedded devices. Home devices must not consume much energy during their operation. Residents would not be willing to change batteries every some days or weeks. We expect that, even in heavy workloads, the batteries should last at least for a few months before total discharge.

- **Acceptable performance in terms of push times**. This concerns requiring minimum time for forwarding events from the application framework to interested subscribers, such as home tenants and third-party Web applications. Even in the presence of numerous simultaneous subscribers (e.g. 20-50), push times must remain less than a second, regardless of the number of events generated by home devices every second.

- **Extensibility**. The system needs to evolve with minimal coupling between all of its components (devices, users, applications etc.). Adding, removing or even updating devices should have minimum influence over the home area network.

- **Scalability in terms of home residents** and requests. The framework must be highly scalable regarding the number of requests from home tenants. It needs to support large numbers of simultaneous users, who may interact concurrently with their home devices. Even though typical families consist of 5-10 members, the application framework should support also

larger numbers of users and requests, e.g. up to 100 requests/minute. This would increase the flexibility of the application framework to be used in the future also in smart building scenarios with numerous tenants/workers. Supporting tens of users covers also the case that each home tenant employs various Web agents to monitor remotely specific aspects of his house, taking clever and informed decisions according to local environmental conditions.

- **Scalability in terms of home devices**. The framework needs to support tens or hundreds of embedded devices. Normally, 10-50 home devices could be available nowadays in some smart home environment, however, perhaps these numbers would change in the future. Hence, a few hundreds of devices should be supported by the application framework.

4.2 Application Framework General Architecture

In this section, an application framework for smart homes is described, based on the requirements defined in Section 4.1. This framework achieves a Web-oriented interaction with the home environment (see Section 2.3), following the principles of REST and resource-oriented architectures (see Section 2.4). The reasoning for selecting these design guidelines is discussed in Section 1.3. We explain below the general design and implementation of the framework, assuming the presence of Web-enabled household appliances (see Chapter 5) and Web clients (home residents), who may interact simultaneously with their smart home environment.

The application framework follows a modular architecture and it is composed of three principal layers: *device layer* is responsible for interaction and management of embedded devices; *control layer* is the central processing unit of the system; and *presentation layer* has the duty of presenting the available devices and their corresponding services to Web users, through their Web browser. In Figure 4.1, the framework's general architecture is illustrated.

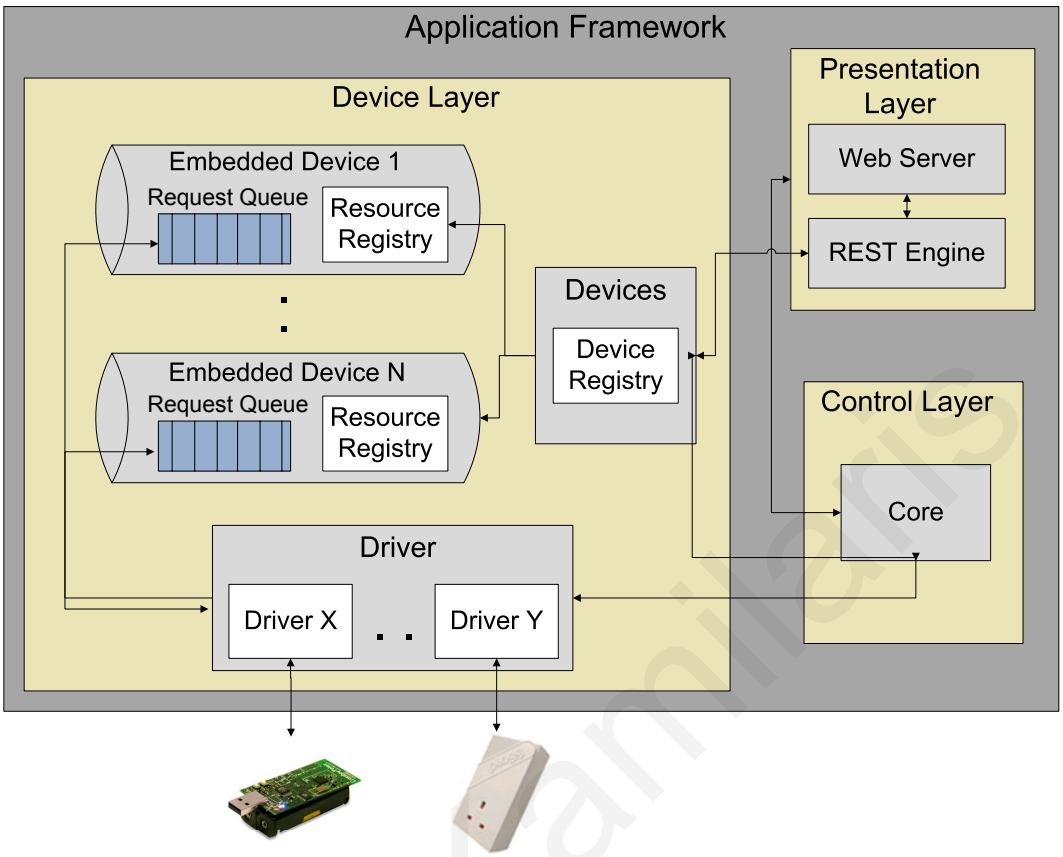

Figure 4.1: Application framework architecture.

In device layer, the *Driver* module holds the technology-specific drivers that are used to enable the interaction with embedded devices. As soon as this module discovers a new local device (see Section 5.2.1), a new thread is automatically created, dedicated to the task of representing the newly discovered device. Each thread keeps track of the static and dynamic properties of the device it represents, such as its health status (through aliveness checks) and a list of the resources it offers, inside a *Resource Registry*. Upon reception of a response from a device through the driver module, this response is forwarded to the appropriate thread, in order to be further forwarded to the Web client, who created the respective request. Using threads for devices simplifies their management, increases their performance and keeps the implementation logic simple.

Hence, after discovery, every operation associated to some embedded device is executed in its own thread environment and behaves independently of its underlined hardware specifications and physical characteristics. Inside each device thread, information is stored concerning device description as well as a list of the services supported by the device. The specific services supported by the home devices that are used in our experiments are presented in Chapter 5, Table 5.1.

Since resource-constrained devices are generally unreliable, a *Request Queue* is integrated to each device thread (see Chapter 6). This queue supports multiple concurrent users, allowing simultaneous Web requests to target the same physical device. Every new request made for a service supported by the device, should be temporarily stored inside that queue. Requests are stored in a FIFO manner (extensions to include priorities are presented in Section 6.4.5) and are transmitted sequentially. As soon as a response arrives in the queue for a previous-made request, a new request is returned from the queue and is transmitted for execution at the device.

If a response does not arrive in a predefined time interval, it is retransmitted to the device in order to mask away possible losses of messages in the unpredictable wireless medium. If after a predefined number of retransmission attempts no response is received, then the embedded device is considered unavailable and it is removed from the system. The time a request will stay in the queue and be executed depends on many factors, such as the device type, its throughput, processing power and network load.

The *Devices* module, also located inside the device layer, is the ”official representative” of the embedded devices to the upper layers of the application framework. Higher layers are directly informed by querying that module about all the necessary information about the devices connected to the system. It maintains a list of all the devices which are bound to the system, as well as all the information covering the services they offer, inside a *Device Registry*. In other words, it offers an abstract way of reaching the home devices by means of a simple API.

The control layer is the soul of the system. It holds the *Core* module, which runs on the background all the time, maintains system's threads, initializes all the framework's modules and checks that everything operates according to its specifications.

The presentation layer represents the access point to the framework from the Web. It is the layer responsible for the interaction with Web users. It is separated from the rest of the system since it is the most extendable part of it. A *Web server* allows Web clients to interact with their smart environment using any Web browser. A *REST Engine* provides REST support to the Web server. The REST engine has been developed using Restlet[1], which is a REST framework for the Java platform. Therefore, the Web clients can interact with any resource through a RESTful API, by means of their Web browser.

Thanks to the links between the resources, tenants can easily explore available devices and their corresponding services via the Web. They can find the desirable devices and service by clicking links, as they would browse Web pages. Figure 4.2 presents a general representation of the devices of the smart home in XML format, inside a Web browser. In the figure, three sensor devices are listed, deployed in different rooms of the smart home (bedroom, kitchen and the living room). Moreover, three smart power outlets are available, connected to a television, a washing machine and a radio. Different representations may be selected (e.g. XML, JSON, HTTP), depending on the needs of the tenants.

The framework is implemented in Java because of the versatility and portability of the language, which allows the framework to run on virtually any device that has a Java Virtual Machine (JVM). It can be considered lightweight as it needs only 2.7 megabytes for its full installation (including needed libraries, drivers and JAR files). Hence, it could be integrated inside almost any computing device of the house, even a router or the mains smart meter.

[1] http://www.restlet.org/

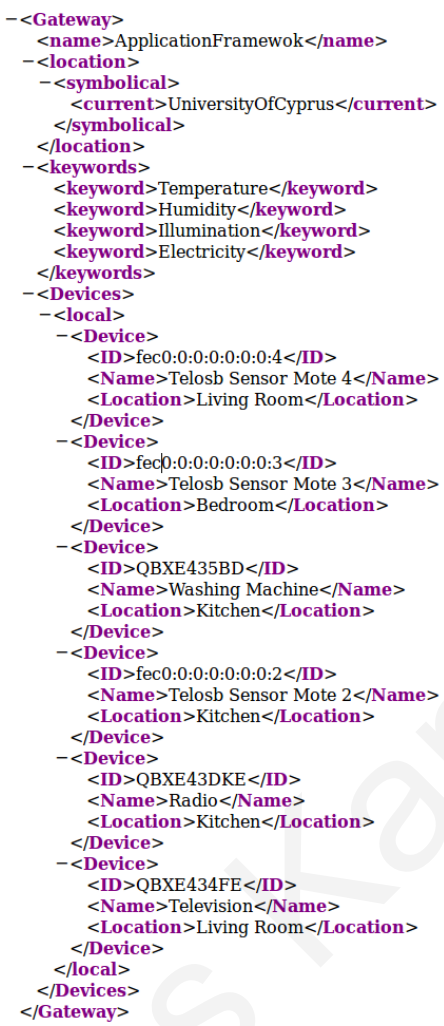

Figure 4.2: A XML representation of home devices inside a Web browser.

The most common system architecture consists of an application framework covering a smart home or a flat. However, in larger-scale indoor scenarios such as a smart building, perhaps an application framework would be responsible for the whole building, in case it could reach effectively all the embedded devices of the building (directly or indirectly through some multi-hop wireless protocol). Another approach would be to place various application frameworks in different computing machines inside the building, allowing the frameworks to interact and collaborate through Web protocols. This possibility is demonstrated in [160].

Appendix B holds the implementation details of the framework, concerning mainly the Java classes of the application. Moreover, Appendix C lists the main parameters that affect the framework's operation and need to be set wisely. The installation procedure of the application framework on some computing machine is described in Appendix D. Finally, the source code of the application framework has been released on the NetRL website[2] and on GitHub[3].

4.2.1 Supporting Various Device Types

The flexible design of the application framework promotes and facilitates the support of various device types and embedded smart home technologies. Heterogeneous home devices only need to implement the methods specified in the driver module, which is located in the device layer of the framework. The general architecture of the driver module is provided in Figure 4.3.

As the figure shows, this module holds the technology-specific libraries that are used to enable interaction with heterogeneous home devices, such as sensor motes, smart power outlets and RFID readers. For each device type, a different software library is added on the driver module, as a Java class that implements the abstract interface *Driver*. Thus, each embedded technology needs to implement the functionality listed in Figure 4.4.

The method *startDevice* performs all the initializations needed to enable communication with some device type, while the *messageReceived* method is responsible for receiving messages from embedded home devices. The two *sendMessage* methods deal with transmitting messages from the framework to the home devices, either in the discovery/description procedure of the device (see Section 5.2) or in invoking some sensing/actuation service offered by the device.

Hence, by implementing these 4 simple methods, the application framework may support any embedded technology for smart homes, including existing popular smart home solutions such

[2] http://www.netrl.cs.ucy.ac.cy/index.php?option=com_content&task=view&id=76&Itemid=54

[3] https://github.com/andreaskami/homeweb

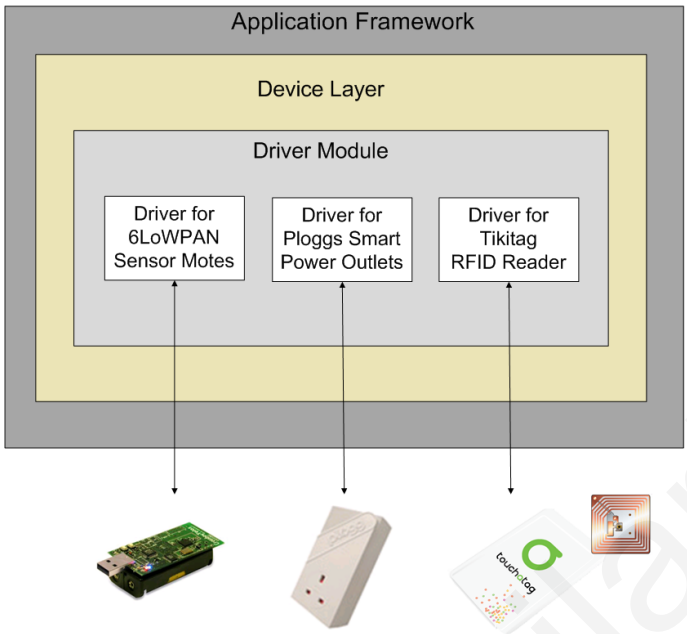

Figure 4.3: The architecture of the *Driver* module.

as X10, KNX, ZigBee, 6LoWPAN, digitalSTROM, Z-Wave etc. Appendix E shows example libraries needed for communicating with Telosb sensor motes and Ploggs smart power outlets (see Section 5.1).

4.2.2 Supporting Multiple Home Devices

To examine the capabilities of the application framework in terms of supporting and managing numerous embedded home devices, we created an emulated scenario, in which multiple virtual devices are assumed to be available inside some smart home environment.

Each device is discovered by the driver module of the application framework, and a new thread dedicated to this newly-found device is created. We experimented with different numbers of devices, considering that the framework has been installed on a typical laptop (Intel Core Duo Dual Core 2.2 GHz). This is actually the computer type on which the framework runs, for the

```
public class NewDeviceType extends Driver{


    // the initializations of NewDeviceType are performed here
    public abstract void startDevice();


    // receive a new message from NewDeviceType
    public abstract void messageReceived
          (String message) throws IOException {}


    // send a message in DISCOVERY and DESCRIPTION mode
    public abstract void sendMessage(
           String id, char type, String content) throws IOException;


    // send a service request in NORMAL OPERATION mode
    public abstract void sendMessage
           (String id, char type, Request request) throws IOException;
}
```

Figure 4.4: The abstract methods that need to be implemented by each home device type, in order to be supported by the application framework.

purpose of the experiments performed in the thesis. For better experimentation, we also installed the framework on a small server (Intel Core i5-3570K Quad Core 3.4 GHz), which is of course a more powerful computing machine. We considered 4 different performance metrics to assess the scalability of the framework in regard to supporting multiple devices. These metrics are:

Table 4.1: Support of multiple home devices on a typical laptop.

Device No.	Disc. Time	Resp. Time Main Page	Resp. Time Devices Info	Memory
10	0.63 s	50 ms	92 ms	10.2 KB
50	5.20 s	726 ms	1119 ms	65.5 KB
100	13.29 s	942 ms	2262 ms	196.6 KB
200	38.17 s	3455 ms	4981 ms	1.7 MB
300	58.80 s	5183 ms	7183 ms	3.8 MB
400	78.16 s	-	-	4.3 MB

1. Device Discovery and Information Parsing Time. This includes the time needed to discover the devices and parse their service description information (see Sections 5.2.1 and 5.2.2).

2. Main Page Response Time. The time needed to load an XML representation of the smart home environment inside a Web browser, listing all the available home devices.

3. Response Time of Devices Page. The time needed to load XML-based service description information of some specific, randomly-chosen home device.

4. Memory. The memory allocated to the application framework during its operation, to support the threads of the physical devices that have been discovered.

The results of our experiment may be observed in Table 4.1 for the laptop case and in Table 4.2 for the server case, considering a number of devices ranging from 10 to 400. Discovery and main page response times are displayed also in Figure 4.5. Obviously, the server machine operates much better than the laptop, giving much better discovery and response times. This improvement in performance is around 30-35% when devices are up to 100 and reaches 50% in some cases, e.g. when devices are equal to 200. The application framework, when installed on the laptop, may

Table 4.2: Support of multiple home devices on a small server.

Device No.	Disc. Time	Resp. Time Main Page	Resp. Time Devices Info	Memory
10	0.81 s	32 ms	55 ms	196.6 KB
50	3.75 s	487 ms	761 ms	262 KB
100	9.11 s	665 ms	1739 ms	458 KB
200	14.82 s s	1659 ms	2470 ms	2.3 MB
300	29.50 s	3507 ms	5730 ms	5.1 MB
400	64.91 s	10604	11924	6.0 MB

support up to 300 devices, with few seconds response times. When installed on the server, it can support up to 400 devices, with response times of around 10 seconds.

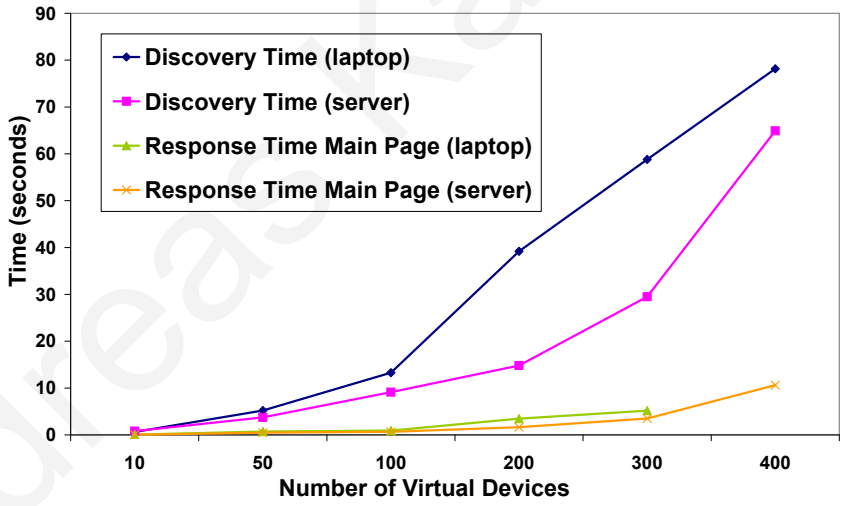

Figure 4.5: Discovery and response times at the application framework in the presence of multiple home devices.

As the number of devices increases, the response times grow as well. As Figure 4.5 shows, the increase in discovery times is almost exponentially, both in the laptop and the server case, while response times grow linearly with an increasing gradient. This increase in response times is partly due to the thread environment, which has quite demanding computational needs in Java. By observing the response times, we consider that the framework installed on a laptop may *effectively* support around 100 devices, and installed on a small server around 200 devices. Larger numbers cause the system to malfunction.

Memory requirements are increased as the number of devices grows, and a few megabytes are needed when 300-400 devices exist in the home environment. These memory requirements are rather small, considering the capabilities of computing devices nowadays.

Finally, the time needed to discover the devices and parse their service description information (see Sections 5.2.1 and 5.2.2) is considered satisfactory, taking into account that around 1 minute time is enough to discover 300 devices in the laptop case and 400 devices in the server case. However, this time is only indicative, since other parameters would influence discovery time in a real-life scenario, as for example the throughput and bandwidth capabilities of the device acting as the base station (see Section 2.1.3). A small real-life experiment considering the discovery time of Telosb sensor motes (see Section 5.1.1) is presented in Section 5.2.1.

4.2.3 Synchronous/Asynchronous Operation

One of the major criticisms against the use of REST in the area of smart homes spans from the fact that HTTP was designed for synchronous requests. This model does not generally suit the requirements of most sensor network applications, including smart homes. However, we managed to overcome this issue by mapping normal HTTP request, coming from Web clients (home tenants)

at the Web server module of the application framework, to an asynchronous operation which happens on the device layer of the framework.

Synchronizers were employed, in order to transform asynchronous behavior into a synchronous one. For each client Web request, the framework creates a new *REST thread*, associated with a unique *message token* (requestID). This message token is encapsulated into the original Web request and the request is forwarded to the thread of the relevant home device. There, it obtains a lock on a unique synchronizer token, labeled using the token.

As soon as the response from the embedded device arrives, the synchronizer lock searches for the request with the respective message token, and the appropriate waiting thread that represents the Web request is awakened. In this way, the HTTP response is then sent back to the initial client. The duration of time needed to complete the Web request is highly dependent of the capabilities of the home device that is invoked, as well as the load at the request queue of the thread that represents this particular device (see Section 6.2). Figure 4.6 shows the sequence diagram for this synchronous/asynchronous translation.

The whole synchronous/asynchronous operation may be summarized as follows:

1. The HTTP request from a resident (Web client) obtains a unique message token (requestID).
2. Together with the message token, the request is sent asynchronously to the thread of the appropriate home device (Device Thread).
3. The request is forwarded from the device thread to the associated physical device.
4. The request obtains a lock on its REST thread, using the token, and puts itself to sleep.
5. The response message arrives at the device thread from the physical home device.
6. The framework awakes the REST thread of the Web request that has the correct token.

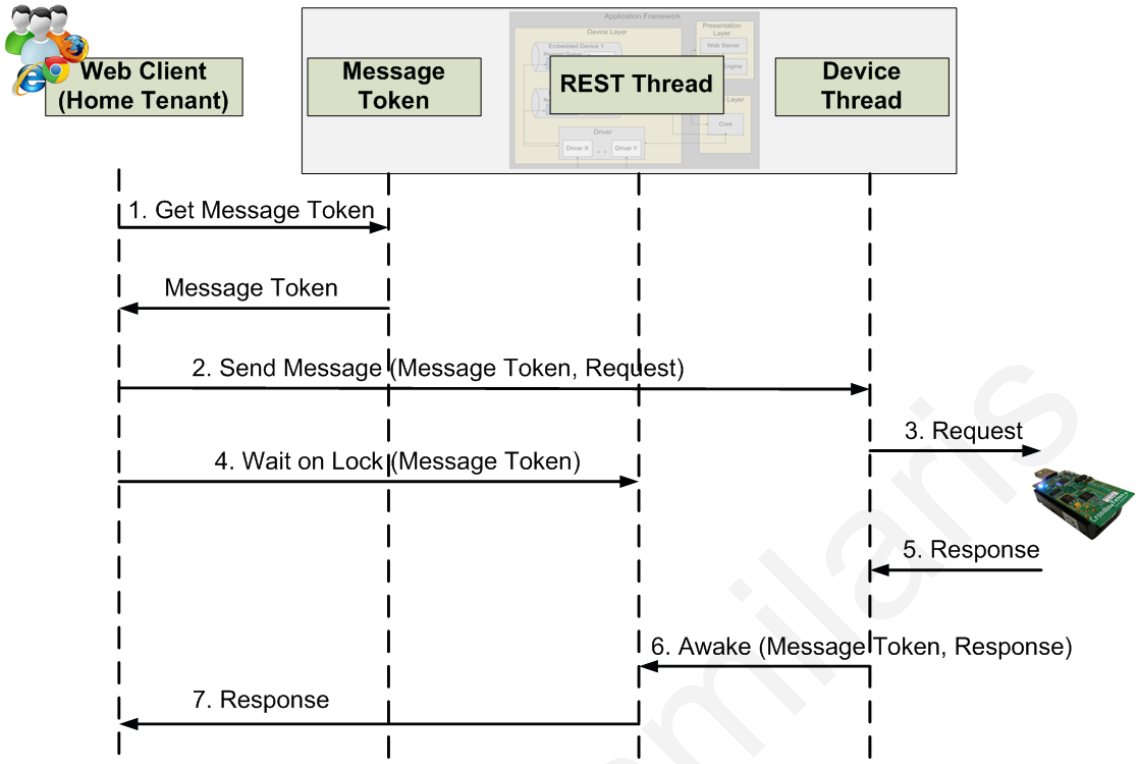

Figure 4.6: Transition between (synchronous) Web operation and (asynchronous) smart home operation and vice-versa.

7. The response is forwarded from the REST thread to the home tenant who created the request.

8. The HTTP response message is forwarded back to the initial resident.

Example code for implementing the sync/async translation is listed in Appendix F.

4.2.4 Handling Failures

Handling device and transmission failures, as well as offering reliability, are important requirements for the application framework (see Section 4.1).

An important mechanism for masking transmission failures are the request queues. Each home device is associated with one request queue, which manages the communication between the physical device and the home residents. These queues have the capability to ”sense” when a transmission failure occurs and retransmit the message back to the device, to avoid message losses. The home device is considered unavailable in case after a predefined number of retransmission attempts no response is received. An analysis of the request queue mechanism and its fast retransmission property, together with potential general benefits, acquired when employing intermediate queues, are presented in Chapter 6.

A mechanism for recognizing fast device failures (which may happen due to empty battery, obstacles in message forwarding etc.) has been developing by performing regular *Aliveness Checks*. These checks are light requests issued to the home devices, in order to determine if they are still alive and operate normally. Through these checks, failed messages due to device unavailability are effectively reduced. A default time interval of 5 minutes is considered for performing these aliveness checks. This time interval must be specified with care: long intervals would reduce the ability for device failure recognition while small intervals would increase the energy consumption of the home devices. Of course, every time a normal response is received from a device, for a previously made service request from a home resident, the framework is confident that the device operates normally and the aliveness timer resets. Hence, aliveness checks are not necessary in cases of high traffic from tenants.

Finally, a mechanism for masking device failures has been implemented, which masks failures by searching for relevant services from other home devices of the same type that exist in the same room of the house, in order to satisfy some resident’s request. In this case, the framework, instead of immediately producing an error message to the Web client who made the request, seeks for another physical device that offers the same service. In case a suitable device is found, then the

request is forwarded immediately to the request queue of that device. When the response arrives, it is forwarded to the corresponding tenant, without him noticing the device failure. This mechanism works only when heterogeneous home devices become interoperable through the use of REST.

4.2.5 Web Caching

Porting the IPv6 stack on sensor devices slightly increases their energy consumption and decreases their performance in terms of response times (see Chapter 7). However, these issues may be minimized by exploiting the HTTP caching feature.

In general, a Web cache is an intermediary between Web servers and clients, monitoring incoming requests and saving copies of the responses for itself. In case there are subsequent requests for the same URL, the cached response is harnessed, instead of asking the origin server for it again.

A lightweight Web cache is included on the application framework. In this case, Web servers are the sensor devices, Web clients are the residents of the smart home and a gateway cache is installed on the framework. The caching mechanism may be used only for GET requests since they are requests that do not alter the state of some resource, according to the REST specifications. The cache uses the expiration model, which is a way for the server to say how long a requested resource stays fresh. *Cache freshness time* defines the validity of some historical sensory measurement, as measured by some embedded home device, in order to employ the cache for satisfying some request, instead of querying the physical device again.

Thus, whenever a client requests a GET service, offered by some specific device which was recently invoked by another client (e.g. the time since this last measurement is less than the cache freshness time), the result is automatically retrieved from the cache, instead of transmitting the request to the physical device again. This is called a *cache hit*, since the measurement is

successfully retrieved from the cache. In case the cache does not contain a fresh answer, this would be a *cache miss*.

The amount of time some measurement is considered valid (cache freshness time) is set on the application framework by some user-defined parameter (see Appendix C). At the technical evaluation in Sections 7.2 and 7.3, this parameter is set to 10 seconds by default. Obviously, a larger amount of time implies more cache hits, however, the results may not be as fresh. On the contrary, smaller cache validity times mean more fresh results but also increased cache misses. These statements are validated experimentally in Section 7.2.2, where increased traffic produces larger percentages of cache hits and decreased response times. The relationship between cache freshness time and response times is also evaluated in this section.

4.2.6 Case Study: Physical Mashups for Smart Homes

The flexibility of the application framework, achieved by following REST principles, facilitates the development of smart applications, in the form of Web mashups. For example, with a few lines of code a database could be created and used, where all the data received from the devices would be automatically saved.

Web mashups are Web resources that include content and application functionality through reuse and composition of existing resources. They can be parallelized with Business Process Execution Language (BPEL)[4], which is the formal language for service composition in WS-* [13]. When Web mashups employ physical devices and services, they can be extended into physical mashups [75]. Physical mashups exploit real-world Web services, offered by physical devices, combining them using the same tools and techniques of classic Web mashups. Our application framework adapts and applies the physical mashup practice in the smart home environment.

[4] http://www.oasis-open.org/committees/wsbpel/

Our framework supports the creation of physical mashups and advanced rules very simply, in any programming language that supports HTTP such as Perl, PHP, JavaScript etc. They may be developed even by inexperienced home residents, who can reuse the functionality of their household information appliances in a uniform and easy way. Physical mashups may be combined easily with Web content, to offer a true Web experience to the real world. For example, residents may share their home devices through online social networking (see Section 8.1) or charge an electric vehicle in low-tariff time periods (see Section 9.1). Tenants may receive an email when the door of their house or their fridge is left open, or they can post to Twitter the song they are currently listening on their Hi-Fi system.

In Figure 4.7, we can observe a function written in JavaScript, in just 13 lines, which implements a locker on a virtual door that only unlocks when the right RFID tag is presented to the *tikitag* device, which is coupled with the framework (located at *localhost* in port *8080*).

Figure 4.8 displays a shell script that implements a distributed rule on sensor motes, in just seven lines of code. This rule checks the temperature of the room, measured by *sensor8* and if it is less than 20 degrees Celsius, then the green LED of *sensor5* is automatically turned on.

To demonstrate how simpler programming becomes by employing physical mashups and REST, we provide in Appendix G the code written in Java, to achieve the same result. In the former case, only seven lines of code are needed, while in the latter, 50 lines of code need to be used. In case the sensors are not IPv6-enabled, this distributed rule would need approximately 200 lines to be executed.

4.2.7 Case Study: Urban Mashups

In order to illustrate more advanced, extended capabilities of the mashup paradigm for the real world, we present below a case study of real-life mashups which target urban environments.

```
function getHttpRequest() {

    var xmlhttp = null;

    xmlhttp.open('GET', 'http://localhost:8080/tikitag/tags', true);


    xmlhttp.onreadystatechange = function() {

        if(xmlhttp.readyState == 4 && xmlhttp.status == 200) {

            var items = eval('(' + xmlhttp.responseText + ')');

            var secret_key = "04:BA:4A:B9:23:25:80";

            for (var i=0; i<items.length; i = i + 1) {

                if (items[i].t == secret_key)

                        $('locked').innerHTML = "Door␣is␣unlocked!";

            }

            if (!unlocked)

               $('locked').innerHTML = "Door␣still␣locked.";

        }

    }

    xmlhttp.send(null);

}
```

Figure 4.7: Sample JavaScript code that checks a number of RFID tags in order to find the right one to unlock the door.

```
function check {
  if [ $? -le 20 ] ; then
   curl -d "color=GREEN" -X POST localhost:8080/sensors/sensor5/Light/
  fi
}

curl -s -X GET localhost:8080/sensors/sensor8/Temperature/ $1
check;
```

Figure 4.8: Sample shell script that implements a distributed rule on sensor motes.

Web-enabled smart homes have the capability to publicly expose the functionality of their sensor devices as open Web API. Home owners could permit this, in order to help other citizens to get better informed about the environmental conditions that exist in the urban environment where they live. This environmental awareness would then assist people to have a better quality of life, by avoiding polluted or highly crowder or even dangerous areas.

Physical mashups can not be directly applied in urban environments, since these dense environments are characterized by high mobility and unpredictability. Sensor devices appear and disappear all the time, without any notice.

Thus, in order to adapt to these highly dynamic landscapes, urban mashups are proposed, defined as opportunistic physical mashups that are validated only when the local environmental conditions support the sensor-based Web services, which are declared by these mashups. Urban mashups are service-centric, meaning that they focus on the services offered by sensor devices and not on the sensors themselves. This happens because a large number of similar sensor devices

may be deployed inside some area, offering the same service. When encountering any of these sensors, the mobile user can be well informed about the service of interest.

Classic Web mashups target advanced functionality on Web sites, achieved by combining Web services. On the other hand, physical mahups deal mostly with the actuation of some physical Web service, triggered when some environmental conditions (defined by relevant Web services) are met. They can well be used in home and building automation scenarios. Urban mashups exploit Web services, offered by sensor devices that are installed in the nearby area, to acquire increased knowledge about the overall local environmental status.

An example urban mashup is displayed in Figure 4.9. In this example, a sensor measuring the levels of noise, a pollution sensor and a street camera combine their services in order to infer the traffic conditions that exist at the center of Zurich.

Figure 4.9: An example urban mashup showing how traffic can be derived from related sensory measurements.

While the mobile user is wandering in some urban area, it is very likely that only partial information about local environmental conditions will be available. Therefore, his urban mashups would not be fully executed, but only validated with some probability.

This probability depends on the effect of each sensory service to the overall inference about the occurrence of some more complex event. Thus, the impact of each physical service needs to be translated into some weight. For example, in the urban mashup of Figure 4.9, noise affects traffic with a weight of 0.4 (intuitively with 40% probability).

Urban mashups can be modeled as a directed acyclic graph with weights, where weights represent the probabilities for the event's actualization. Extending this directed graph, more complex relationships can be created and automatic inference about more advanced environmental conditions can be achieved. For example, traffic combined with weather conditions could trigger the suggestion (to some citizen) of using public transportation to go to work. This example is illustrated in Figure 4.10, assuming that traffic affects bus recommendation with weight of 0.6 while the weather conditions have an influence of 0.4 to the validation of this mashup. A user-defined *probability threshold* decides whether we should notify the user about a possible triggering of a mashup, which corresponds to some event. This threshold relates to the sensitivity associated with some urban mashup triggering.

A number of relevant parameters influence the effectiveness of the inference process. These parameters can be the distance of each sensor device from the mobile user, the altitude of each sensor from the point of interest, the accuracy of each device and the number of sensors offering the same service. This last parameter can be parallelized to GPS services, whose accuracy depends on the number of available satellites.

Since urban mashups are suitable mostly for mobile users, it is more appropriate to be supported by mobile applications. Thus, mobile applications could support urban mashups, in order

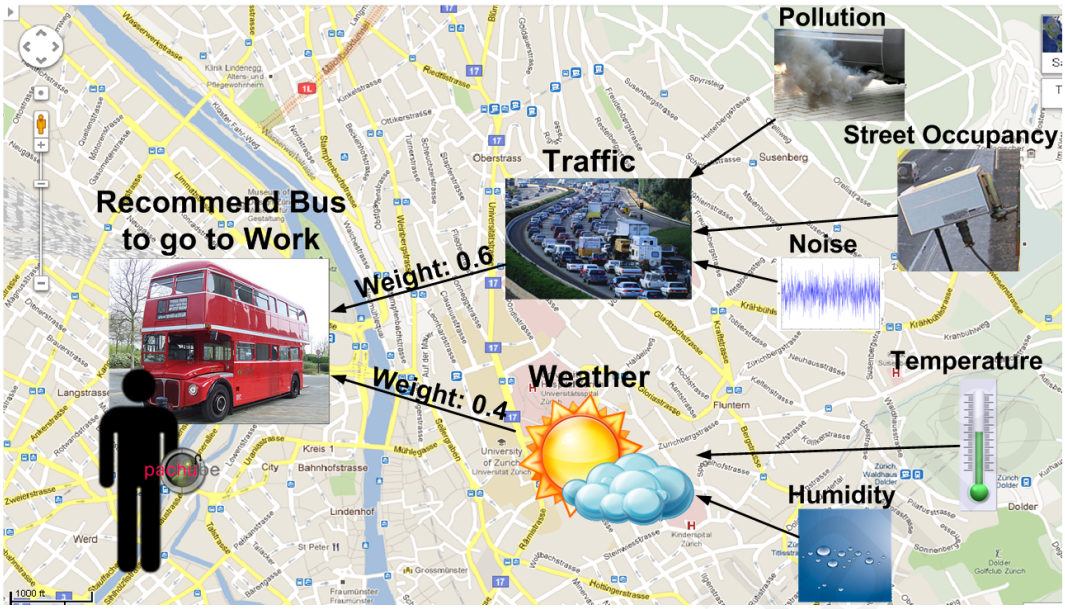

Figure 4.10: An extended urban mashup demonstration, recommending bus to go to work.

to assist people in their everyday lives. We will discuss the development of such an application, called UrbanRadar, in Section 10.4.2.

Undoubtedly, supporting urban mashups is not a trivial task, because the sensor devices supporting environmental services need to be globally discovered in real-time, through the Web. The WoT has not yet defined any protocols for automatic discovery of Web-enabled physical services. In other words, a search engine for the physical world is currently unavailable. Dyser [126] and Snoogle [172] are two early approaches proposed for real-time search for physical entities such as sensor devices. However, they need a long way before they become efficient, popular and massive engines for the real world. A novel pervasive technique for automatic discovery of Web-enabled environmental services is proposed in Section 10.3.

Until the Web community adopts a scalable, efficient approach for global discovery of envrironmental services, the UrbanRadar mobile application (see Section 10.4.2) harnesses online, large-scale sensor directories to achieve urban environmental awareness.

4.3 A Graphical Application for Smart Homes

In this section, we describe the extension of the Web-oriented application framework into a Web-based graphical application for smart homes, where all interactions with embedded devices are done via standard HTTP requests. We explain its extended architecture as well as the graphical user interface (GUI) we implemented for facilitating the interaction with the embedded devices of the home environment.

4.3.1 Extended Architecture

To enable a true Web-based smart home application, the architecture of the application framework was extended to support a graphical application. The classical client-server model was followed, by placing the framework's functionality on the server side and the graphical application on the client side. The extended system can be observed in Figure 4.11.

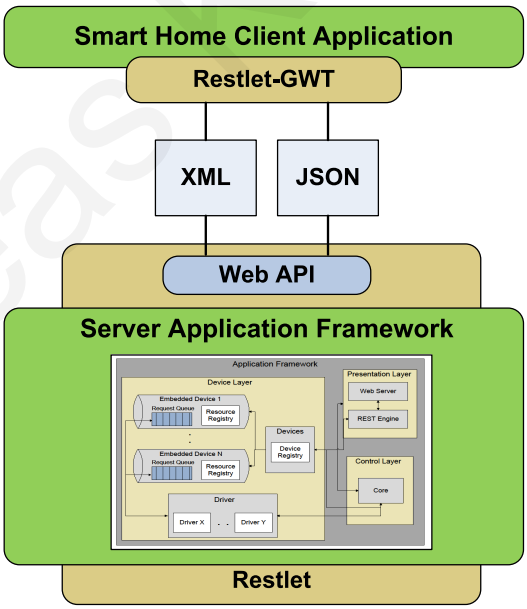

Figure 4.11: Extended architecture of the Web-based smart home.

The client application offers a Web-based, interactive GUI in AJAX, in order to abstract the home automation procedure. It is developed with Google Web Toolkit (GWT)[5]. Graphs that illustrate sensory and electrical consumption data are produced by means of Google Chart Tools[6].

Client-server communication is based on HTTP calls. The application framework exposes its RESTful interface as a Web API, which is utilized by the client application to interact with the embedded devices of the smart home. Response messages from the server are encapsulated in open, well-known formats such as JSON and XML.

4.3.2 Interfaces of the Smart Home Application

The client application is split into various graphical interfaces. *Devices* interface lists the available sensor devices, *Services* presents the services offered by some home device while *Request Builder* is an easy way to create Web-based requests for a 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 hook[7] (see Section 2.6.4). For example, the resident may program the sending of an email as soon as the humidity in some room of the house exceeds 60%.

Furthermore, *Physical Mashups* interface allows the easy creation of mashups in a few clicks e.g. to switch off the air conditioner, when the temperature reaches 25 degrees Celsius (see Section 4.3.4). *Task Scheduling* allows the automatic execution of some service in a future time. This interface is used in Section 9.2 to exploit the demand response program of a virtual smart grid and program the execution of electricity-related tasks in hours of the day with low tariffs, as these are defined by electric utilities in order to balance the total energy demand.

[5] http://code.google.com/intl/en-EN/webtoolkit/

[6] https://developers.google.com/chart/

[7] www.webhooks.org/

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

Figure 4.12 shows a snapshot of the graphical application, in which the services offered by a particular device are presented. In this case, a Telosb sensor mote (see Section 5.1.1) is shown, offering four different services: temperature, humidity and illumination sensing services; and actuation of its three LEDs. The user can view the current state of each service just be hovering over it with the mouse pointer. In the figure, the current temperature value is displayed.

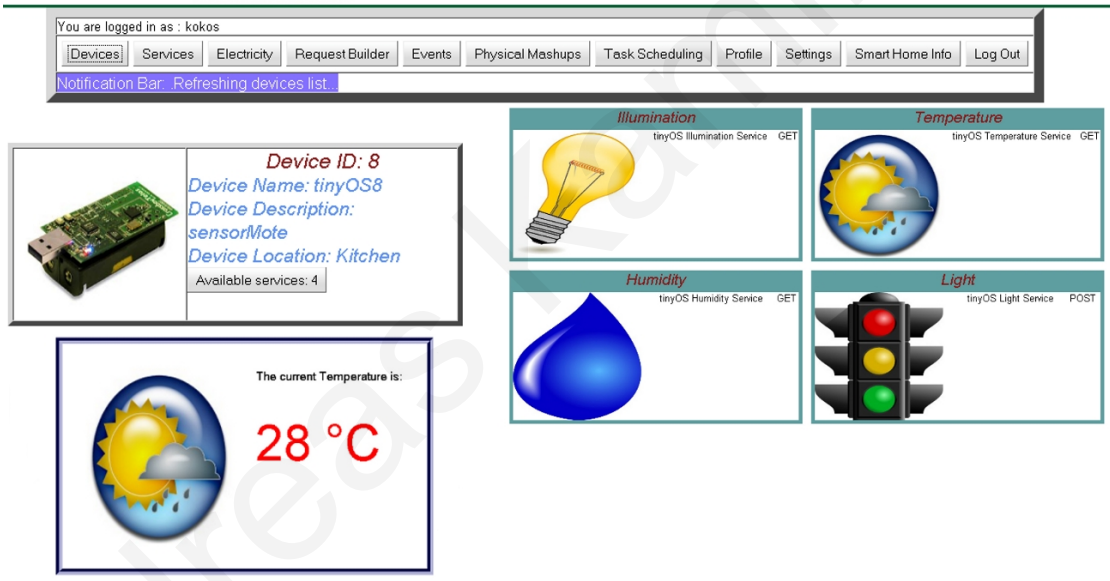

Figure 4.12: Snapshot of the Web-based smart home application.

4.3.3 Case Study: Energy Awareness through Real-Time Feedback

By understanding the electricity footprint they consume, residents will be able to make intelligent choices about energy. Through detailed energy monitoring, electricity-wasting actions can be

avoided and energy-inefficient devices can be managed better or be replaced. We further adapted our smart home application to allow residents to view their electricity footprint via a Web interface. We employed smart power outlets (see Section 5.1.2), associated with the electrical appliances of the house, and we exposed their functionalities as RESTful Web services (see Chapter 5, Table 5.1, Resources 5-6) to develop a physical mashup in combination with Google Chart Tools.

Through the Web, home residents can switch off electrical appliances, if they forgot them on when they hastily left the house. Residents can associate the energy consumption of their electrical appliances with the actual tariffs from their electric utilities, translating kilowatt hours (kWh) into money. In such a way, the user is not confused with difficult to understand measurement units.

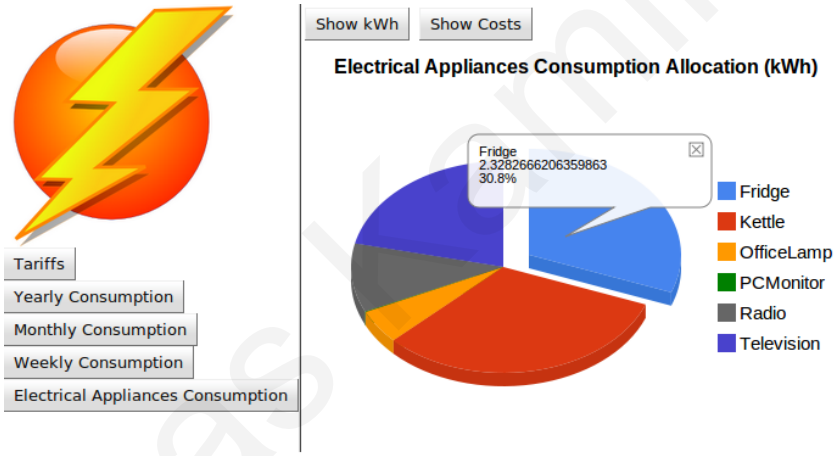

Figure 4.13: Detailed electrical consumption distribution of home's electrical appliances.

Historical data about energy and costs are aggregated into graphs that present useful remarks about the yearly, monthly and weekly consumption in the whole house or per appliance. By comparing their actual electricity footprint with previous ones, citizens can be aware in real-time if their actions are energy- and cost-efficient. By determining the impact of their actions in a timely fashion, they can evolve a sustainable, greener behavior. Figure 4.13 presents a typical snapshot of

the client graphical application, in which the distribution of the energy consumption and the costs of the electrical appliances in a typical home are presented in a pie chart. Figure 4.14 denotes the electricity footprint of individual household electrical appliances on a daily basis for the current week. Monthly and yearly view of historical information is also available.

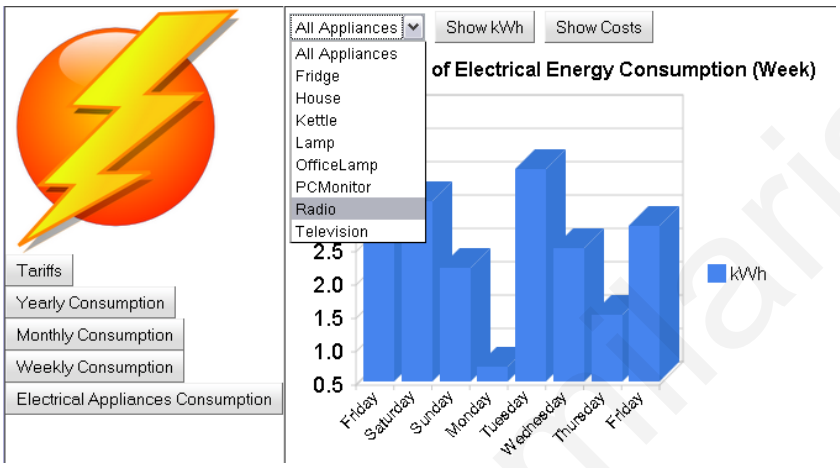

Figure 4.14: Detailed electrical consumption historical information of electrical appliances.

To further illustrate the flexibility of Web-enabling smart meters, we adapted the *EnergieVisible* project [43], to operate in accordance to our smart power outlets. EnergieVisible is a mashup developed by GWT 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. Since the operation of EnergieVisible includes polling continuously data from a RESTful server, we simply needed to point it to our application framework. It works directly as long as devices offer their data using the correct JSON syntax. The EnergieVisible application, connected in real-time to our application framework and the smart power outlets, is shown in Figure 4.15.

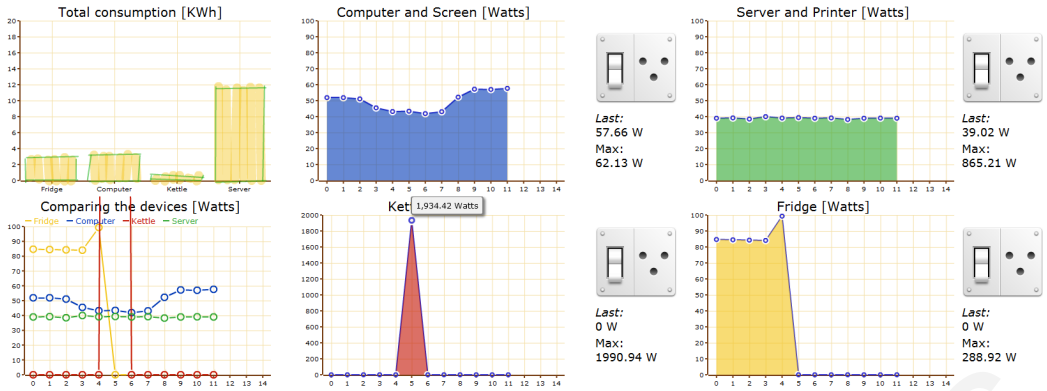

Figure 4.15: The *EnergieVisible* Web interface [43] connected to our application framework.

4.3.4 Case Study: Energy-Efficient Smart Rules

When the resident receives timely information about the environmental context of his home (e.g. temperature, light, motion), together with real-time data of the consumption of his electrical devices, he can take clever decisions to control their operation in an automated, energy-efficient way. Offering to a resident the means to actively take part in energy-saving initiatives could help him to substantially decrease his overall consumption [138]. Each person has unique behavioral patterns and the potential to configure his home energy usage effectively to suit his needs.

To promote energy efficiency, we developed a graphical mashup editor, through which a resident can create his own smart rules to automate his house in the form of physical mashups in just a few clicks, using operations such as greater than, if/then, and/or, etc. He can combine environmental sensory measurements with services offered from his electrical appliances, such as sensing in real-time their energy consumption and switching them on/off (see Section 5.2.3). Each habitant maintains his own profile, being able to view, edit and delete his mashups.

A snapshot of the mashup editor is provided in Figure 4.16, demonstrating an example mashup. It is assumed that a resident has the bad habit to turn off the lights when he goes to sleep but to

forget the television and the DVD player on. As soon as a sensor device placed in the living room senses reduced illumination, then the television and the DVD player are automatically turned off. By creating this mashup, this tenant can avert this habit, which costs him energy and money.

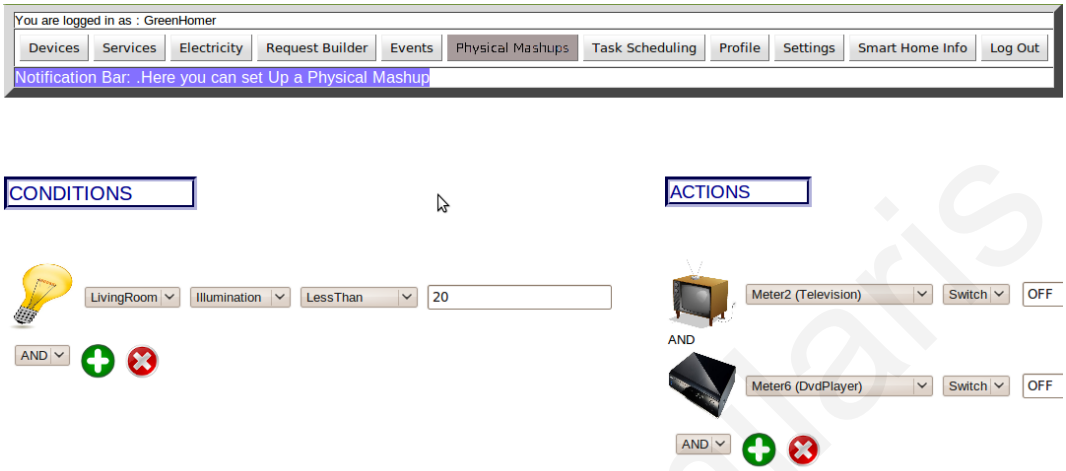

Figure 4.16: An example of a physical mashup at the physical mashups editor.

More advanced mashups could also reveal undesirable events such as electricity leakage or unexpectedly high temperatures, and countermeasures can be automatically taken, such as to turn off immediately faulty appliances and minimize the possibility of fire.

Home appliances, when exposed as Web resources (see Chapter 5), can be combined seamlessly with content and services provided through the Web. The mashup editor supports the blending of the smart home with the Web. For example, the user can employ a weather forecast service, provided by a reliable organization through the Web (e.g. World Weather Online[8], Yahoo Weather[9]), to adjust the temperature of his house in a future time. In case the outside temperature is predicted to substantially increase after some hours, the user can exploit this information to turn off the electric heating automatically before this increase happens.

[8] http://www.worldweatheronline.com/

[9] http://weather.yahoo.com/

Chapter 5

Web-Enabling Home Devices

The WoT envisions Web presence for physical things in general and home devices in particular. According to this vision, home devices are supposed to offer plug and play functionality, operating seamlessly in the home environment as soon as they are turned on, whether they are stationary or mobile, indoor or outdoor. Standardized methods and techniques need to be used for discovering these devices, describing their services, communicating with them, even sharing them and their services between family members, friends or colleagues.

Moreover, embedded devices could form a mesh network in the smart home, where messages would be propagated through multi-hop wireless communications between the devices. Smart home frameworks and applications could interact with them in an ad hoc manner, with minimal prior knowledge about them, without requiring to install drivers or specific applications. Finally, home devices shall expose their services in well-understood, easily accessible, clean and open interfaces, facilitating their integration in smart home applications, which could access them in a uniform and standardized way. In the following sections, we show how we suggest establishing Web presence for home devices, addressing the requirements posed by the general concept of the WoT, applied specifically for smart home environments.

The rest of the chapter is organized as follows: Section 5.1 presents the embedded devices that were employed in our smart home deployment and experiments, namely Telosb sensor motes and Ploggs smart power outlets. Then, Section 5.2 discusses various issues encountered during the process of Web-enabling these resource-constrained devices. The approaches used to address these issues are explained with regard to the general operation of the application framework for smart homes (see Section 4).

5.1 Embedded Devices for the Smart Home

The embedded devices involved for enabling a Web-based smart home include sensor motes and smart power outlets. These devices are employed in the experimental deployment of a smart home, used for analysis and evaluation reasons in Chapters 6 and 7, as well as in Chapters 8, 9 and 10, during various case studies that demonstrate the potential of Web-enabled smart homes. These two device types are presented in the following subsections.

5.1.1 Sensor Motes

Telosb sensor motes [134] are equipped with a 250kbps, 2.4GHz, IEEE802.15.4[1]-compliant Chipcon CC2420 Radio, integrated onboard antenna and a 8MHz TI MSP430 microcontroller with 10 kB RAM. These motes integrate by default temperature, humidity and illumination sensors, but can also support other sensors (e.g. noise, pollution, electromagnetism, occupancy). A Telosb mote is shown in the left side of Figure 5.1.

[1] http://www.ieee802.org/15/pub/TG4.html

The sensor motes were programmed using TinyOS 2.x[2], which is an operating system designed for low-power wireless devices (see Section 2.1.4). By means of Blip[3], which is an implementation of the 6LoWPAN stack for TinyOS, the default sensing capabilities of the motes were exposed as RESTful Web services, transforming them into embedded Web servers. Hence, their sensing services can be accessed using standard HTTP requests under a RESTful interface.

Through 6LoWPAN (see Section 2.1.6), an IPv6-enabled multi-hop WSN may be formed, consisting of Telosb motes. Thus, by deploying these devices inside the smart home, the environmental conditions inside the house can be sensed in real-time, while the multi-hop topology would enable the wireless propagation of the measurements from the remote sensors to the base station.

In the case of our Web-based smart home environment, the base station would be attached to the computing machine that hosts the application framework for smart homes. Guidelines concerning installation of the TinyOS software on the Telosb motes are provided in Appendix H.

5.1.2 Smart Power Outlets

Ploggs are smart power outlets that sense the energy consumption of electrical appliances. We selected Ploggs[4] as the power outlet devices in our smart home deployment as they have a high accuracy of electricity measurements, with an average error around 2%, according to [111].

A Plogg can be attached to any electrical appliance or device that uses a standard mains socket plug. Its hardware is based on Ember's EM357 ZigBee wireless technology and Teridian's 6511 metering chip. They use a proprietary firmware based on the ZigBee protocol, being capable of forming a mesh network. Therefore, through the ZigBee chip, Ploggs can create a wireless, multi-hop smart metering network inside a smart home.

[2] http://www.tinyos.net/

[3] http://smote.cs.berkeley.edu:8000/tracenv/wiki/blip

[4] http://americannewsreport.com/plogg-mains-plug-measures-real-consumer-use-of-household-appliances-889901

They can transmit with high accuracy energy consumption measurements in a user-defined interval, with one minute as the default value. Furthermore, they allow the control of a connected electrical appliance remotely, by switching it on/off. External transducers can be factory-fitted to Plogg devices to provide whole-home consumption measurements, by plugging them to the mains supply. We can observe a Plogg smart power outlet in the right side of Figure 5.1.

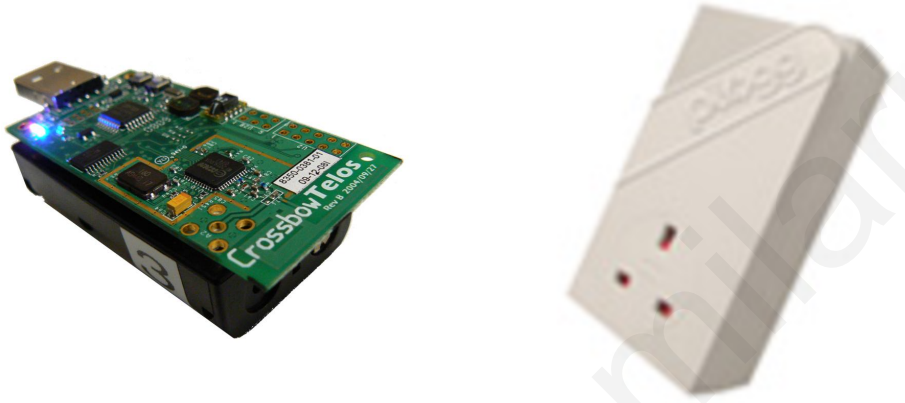

Figure 5.1: A Telosb sensor mote (left) and a Plogg smart power outlet (right).

Since Ploggs are programmed with a firmware that can not be easily changed, they may not be enabled directly to the Web. Thus, they are enabled indirectly, through the application framework, by developing Java drivers to communicate with them through a REST interface. Nonetheless, we estimate that future generations of smart meters could actually run a full Web server on-board, thus enabling direct, non-mediated interaction with other peers in the same WSN, such as sensor motes, forming a single Web-based multi-hop wireless network. ACme [93] is an early effort towards this vision.

5.2 Addressing Issues for Web-Enablement of Devices

The task of enabling physical devices, sensors, actuators, smart meters and smart power outlets to the Web, has various issues that need to be addressed. At first, the local discovery of these

devices in some pervasive environment such as a smart home needs to be performed. Then, the capabilities and the functionality offered and supported by the devices must be described in a standardized way, to facilitate service invocation by third-party applications. After, the interaction possibilities with the devices shall be enabled, using uniform and standardized approaches. Interaction could follow a request-response manner, similar to the classical client-server model, or obey to a push-based technique, to support event-based scenarios with events triggered only sporadically. Finally, the possibility of sharing Web-enabled home devices and services between family members, friends or colleagues through the Web is also considered.

5.2.1 Local Device Discovery

A crucial aspect in embedded computing involves the discovery of physical devices in some physical environment. The adopted local device discovery protocol was motivated from WS-Discovery[5], which is a multicast discovery protocol that uses SOAP/XML messages to locate WS-* services on a local network.

This protocol was adapted 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. Home devices declare their presence in the home environment in frequent time intervals (e.g. every 30 seconds), until they are bound to some smart home application, such as our application framework for smart homes. The idea of associating Uniform Resource Locators (URLs) with physical devices for describing their functionality was first mentioned in [103].

When an embedded device joins the IPv6 network, it announces its existence through the multicast message. The application framework listens at the pre-specified multicasting channel

[5]http://docs.oasis-open.org/ws-dd/discovery/1.1/os/wsdd-discovery-1.1-spec-os.html

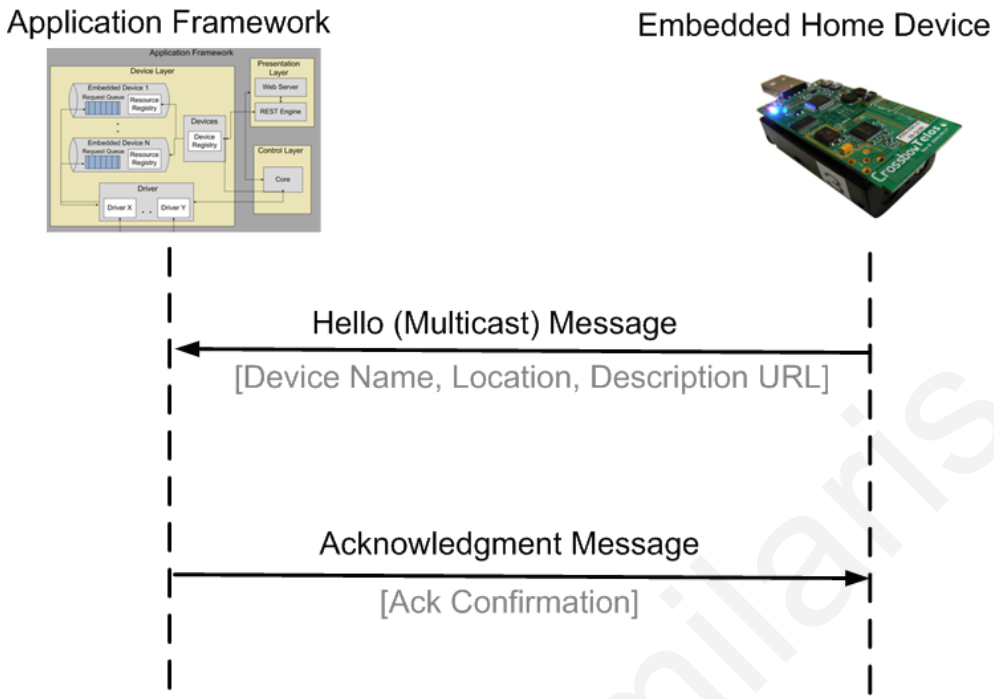

Figure 5.2: Device discovery protocol.

for incoming discovery requests. As soon as it receives a new request, it responds to the newly found sensor device by an acknowledgment message. Then, the framework creates a new thread, dedicated to this device, and updates the device registry. On the other hand, when the device receives the acknowledgment, it begins to operate normally as an embedded Web server, accepting requests from the framework. Figure 5.2 depicts this device discovery process.

Through this discovery protocol, a plug and play approach for automatically including home devices in the smart home is achieved. Figure 5.3 provides the end-to-end time needed at the device discovery procedure, from the time when multiple devices announce simultaneously their presence at a smart home environment until they are registered to the application framework and an acknowledgment confirmation message is transmitted to each device.

In this small experiment we performed, the devices were deployed in a star topology, 5 meters distance from the application framework. Less than a second was needed from the time some sensor mote was switched on, until it was discovered by the framework, even when 10 sensor motes were switched on at the same time. This indicates that our local device discovery mechanism is scalable and operates with satisfactory performance.

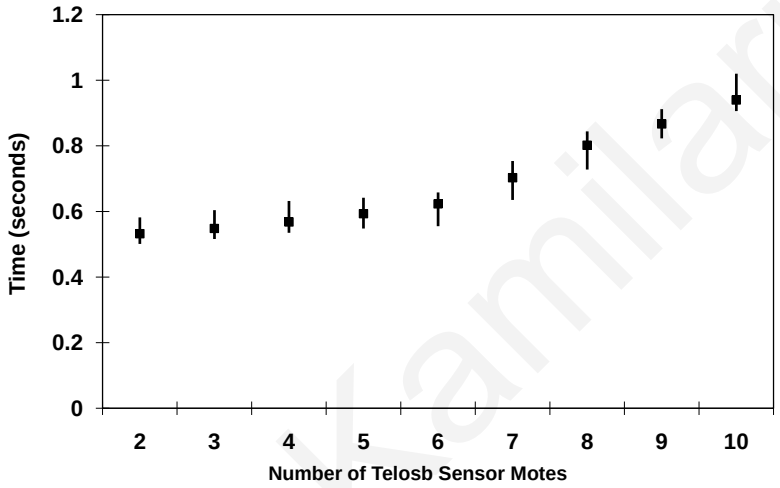

Figure 5.3: Device discovery with multiple simultaneous Telosb sensor motes.

The proposed discovery protocol is actually a lighter version of WS-Discovery, which is used in WS-* for locating local services. By transmitting a single URL that points to the service description information of the device, instead of a heavy SOAP/XML message, our protocol becomes much more efficient and scalable.

Our proposed protocol looks similar to Multicast DNS (mDNS) [29], which is a protocol that provides the ability to perform DNS-like operations on a local network in the absence of any conventional unicast DNS server. mDNS uses API similar to unicast DNS but implemented over

a multicast protocol. Our protocol operates in a similar way, based on multicasting, however it is simpler and as reliable as mDNS. Instead of using DNS-like operations, only a URL is transmitted through a HTTP message, which describes the services of some local device in a standardized way (see next section). A similar protocol to mDNS is DNS-based Service Discovery (DNS-SD) [28], which proposes using standard DNS programming interfaces, servers and packet formats to browse a network for services. Specifically, it uses DNS SRV, TXT, and PTR records to advertise Service Instance Names. Our protocol is similar also to DNS-SD, avoiding the DNS-based actions by sending a single URL.

More complex protocols that could be adopted include Simple Service Discovery Protocol (SSDP) [64] and Service Location Protocol (SLP) [80]. SSDP is a network protocol based on IP for advertisement and discovery of network services and presence information. It is the basis of the Universal Plug and Play (UPnP) discovery protocol and is intended for use in residential or small office environments. Services are announced by the hosting system with multicast addressing to a specifically designated IP multicast address and port number. SSDP is a HTTP-like protocol and works with NOTIFY and M-SEARCH methods. We argue that these methods violate the RESTful environment we propose.

SLP is another service discovery protocol for local networks using URL schemes, attributes and service types, as well as query types to locate services. SLP has three different roles for devices, depending on whether they announce or search for services. SLP also employs multicasting for locating services. We believe that SLP adds to the total complexity of the home ecosystem, by using protocol-specific schemes to describe services. Our protocol performs a similar job with less complexity, following merely principles of the well-known HTTP protocol.

5.2.2 Service Description

In general, embedded devices need to describe their functionality in order to allow the invocation of their services by third-parties. In a smart home scenario, home devices need to describe their services to the application framework, to enable seamless interaction with family members, through the Web. Service description must be performed in a standardized, Web-based way, to increase the interoperability between heterogeneous service providers and consumers.

We adopted Web Application Description Language (WADL)[6], which is an XML-based file format that provides a machine-readable description of HTTP-based Web applications. 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. It can be seen as the RESTful equivalent of Web Services Description Language (WSDL)[7], which is the standard for describing SOAP-based WS-* Web services [13]. WADL is considered a Web-based method to perform resource description, in order to achieve high interoperability between heterogeneous home devices and services. WADL promotes application reuse, since it constitutes a flexible way of describing Web services.

WADL is used to model the services (resources) provided by embedded home devices, and the relationships between them. Each home device maintains its own WADL file in some URL, which could be specified by the device manufacturer. Obviously, all devices of the same type, may share the same WADL file. A device encapsulates the link to its WADL file in the URL that is transmitted to the application framework in the local device discovery procedure (see Section 5.2.1).

[6] http://wadl.java.net/

[7] http://www.w3.org/TR/wsdl

Example WADL files for the Telosb sensor motes and Plogg smart power outlets, are shown in Appendix I. Figure 5.4 displays a WADL description file for a Plogg smart power outlet inside a Web browser, through the application framework. In the figure, this Plogg is associated with a washing machine and offers two services: real-time feedback of the electrical consumption of the appliance and the capability of switching it on/off. Whenever a Web user (e.g. home resident) wishes to interact with some device of his smart home, he only needs to follow the guidelines specified at its WADL file. In this way, automatic M2M interaction may also be supported.

As soon as the framework discovers a new device, it parses the discovery message transmitted by the device to find the URL that points to the associated WADL file. By parsing the WADL file, the framework can be informed about the services offered by the device, as well about how to invoke them (e.g. URIs, parameters, return values). This information is then stored at the resource registry of each device's thread.

An emulated scenario that considers the time needed to discover and parse the WADL-based service description information of hundreds of *virtual* home devices by the application framework was presented in Section 4.2.2. In this experiment, the main source of delay was the time needed to parse the WADL files of the home devices. However, this delay was very small, since 1 minute time was enough to discover 300 devices in a laptop case and 400 devices in a server case. Hence, the service description procedure scales very well with hundreds of simultaneous home devices.

Relevant languages for describing sensor information include Sensor Model Language (SensorML)[8] and Extended Environments Markup Language (EEML)[9]. However, these languages focus mostly on describing sensory measurements, and not on explaining how to invoke some particular service offered by a physical device.

[8] http://www.opengeospatial.org/standards/sensorml/

[9] http://www.eeml.org/

```
−<device>
  <name>Washing Machine</name>
 −<location>
  −<symbolical>
    <current>/</current>
  </symbolical>
 </location>
 −<keywords>
    <keyword>Electricity</keyword>
    <keyword>Switch</keyword>
 </keywords>
 −<resources>
  −<resource>
    <name>Switch</name>
    <description>Plogg Smart Meter Switch Service</description>
   −<methods>
    −<method>
      <name>POST</name>
      <description>Set the electrical appliance ON/OFF</description>
     −<mimetypes>
        <mimetype>text/plain</mimetype>
      </mimetypes>
     −<parameters>
      −<parameter>
        <name>mode</name>
        <description>Switch on/off the appliance</description>
        <type>xsd:string</type>
        <option>On</option>
        <option>Off</option>
       </parameter>
      </parameters>
     </method>
    </methods>
  </resource>
 −<resource>
    <name>Electricity</name>
    <description>Plogg Smart Meter Electricity Service</description>
   −<methods>
    −<method>
      <name>GET</name>
      <description>Measure electrical consumption in Watts and kWh</description>
     −<mimetypes>
        <mimetype>application/json</mimetype>
      </mimetypes>
      <parameters/>
     </method>
    </methods>
  </resource>
 </resources>
</device>
```

Figure 5.4: A WADL description file for a Plogg smart power outlet.

Microformats[10], as a set of simple, open data formats built upon existing and widely adopted standards, are simple ways to add information to a Web page by embedding semantic information using mostly the *class* attribute of the HTML language. Search engines and Web parsers can then parse these sites and get informed about the semantics of information. Microformats could be

[10]http://microformats.org/

Table 5.1: RESTful services offered by Web-enabled embedded home devices.

No.	Device	Resource	Description	MIME	Verb	Parameters
1	Telosb	Temperature	Measurement in Celcius	text/plain	GET	-
2	Telosb	Humidity	Percentage of humidity	text/plain	GET	-
3	Telosb	Light	Set RGB LEDs on sensor board	text/plain	PUT	color (char)
4	Telosb	Illumination	Illumination levels in Lux	text/plain	GET	-
5	Plogg	Electricity	Instant consumption in Watts	JSON	GET	-
6	Plogg	Switch	Switches an appliance on/off	text/plain	PUT	mode (String)

possibly used in the future, for description of devices and services on the Web. However, they are more suitable for the global discovery and description of Web-enabled devices (see Section 10.3), and not their local discovery/description in smart home environments.

5.2.3 Request-Response Interaction

The typical interaction pattern between home residents and physical devices is the request-response model, similar to the classical client-server model used at the Web. Since the principles of the WoT are followed and REST is proposed for interaction with embedded devices, the services offered by these devices may be modeled as Web resources, identified by URI. Resources can only be manipulated by the methods specified in the HTTP standard (e.g. GET, PUT, POST, DELETE), under a uniform interface.

Concerning the embedded devices employed in our smart home deployment (see Section 5.1), namely Telosb sensor motes and Plogg smart power outlets, their capabilities and functionalities are exposed as RESTful Web services, to facilitate the uniform and standardized interaction between home devices and smart home applications, such as our application framework. This REST-based modeling of the services of home devices is provided in Table 5.1.

5.2.4 Event-Based Messaging

Since home devices (mainly Telosb sensor motes) operate as tiny 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 means that the application framework must ask the devices in frequent intervals about their measurements. 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 triggered only 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 home device in our case. Push technique is inspired from Web-based publish/subscribe communication.

To support event-based messaging, we build upon recent developments in Web push techniques (see Section 2.6) and extend them for embedded devices with a RESTful messaging system, in order to efficiently integrate event-driven services to our smart home environment. We use the RESTful Message System (RMS) [161], which is a lightweight publish/subscribe messaging suited for embedded devices. RMS proposes a lightweight alternative for messaging, suited for home devices that monitor the home environment for events. It is a push technique, implying that communication is initiated by the physical devices and not by a smart home application.

RMS is applied to the IPv6-enabled sensor motes, by enhancing them with a simple RESTful API to use the eventing system. In case the application framework (subscriber) wants to receive notifications about an event from a sensor mote (publisher), it creates a new subscription by POSTing an HTTP request to the mote. 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. Hence, devices inform the framework only when something important happens. Receiving events in real-time through user-defined callbacks over HTTP, is the notion of Web hooks (see Section 2.6.4).

5.2.5 Device/Service Sharing

Sharing of Web-enabled embedded devices and pervasive services between family members, friends or colleagues is an important issue. We argue that this can be achieved by harnessing online social networking sites. Blending online social entertainment with control of the home environment can give incentives to people, for adopting sustainable lifestyles.

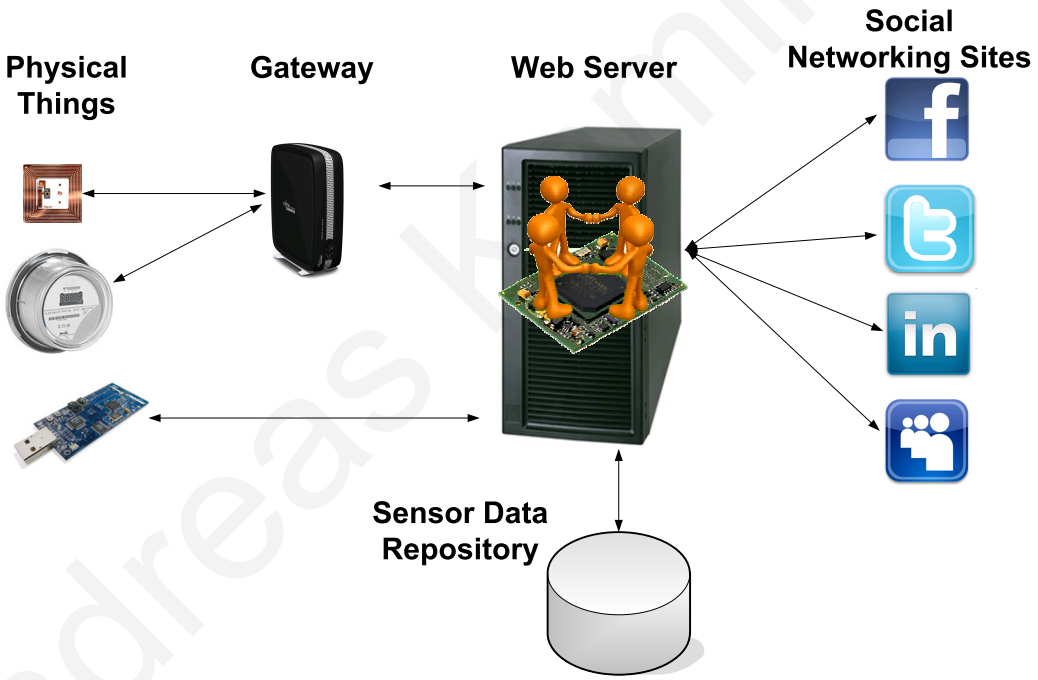

Figure 5.5: General architecture for developing pervasive social networks.

Since the application framework exposes its functionality and the home devices under a RESTful API, it is easy to develop physical mashups including the household appliances of the smart

home inside a social networking application. The Web 2.0 technologies provided by social networking sites can be utilized to transform the interaction with the smart home into a shared, social experience. The general architecture for enabling service sharing of home devices through online social networking is presented in Figure 5.5. It consists of the following three components:

- The physical devices that are Web-enabled, directly through embedded Web servers or by means of gateways. The devices/gateways offer RESTful Web services for interaction with them (see Section 5.2.3) and Web-based device discovery and service description patterns (see Sections 5.2.1 and 5.2.2). These devices can be sensors, actuators, RFID tags, smart meters, smart power outlets etc.

- The social networking sites (SNS), which provide Web-based open API, supporting the application development from third-party developers. Pervasive SNS applications are developed following the mashup example, extending Web mashups into physical mashups, enabling home devices to be integrated to the social structures of the SNS.

- A Web server (e.g. our application framework for smart homes) that hosts pervasive SNS applications, maintaining at the same time connectivity with physical devices/gateways. The Web server needs to have a precise view of the smart home environment it represents, in order to automatically add/remove ad hoc devices that appear/disappear frequently.

In certain occasions, especially when sensor devices are involved, their sensory readings can be automatically stored in some repository. In these cases, WS-* can also be used, for interaction between the Web server and the repository.

By utilizing Web services, interoperability with most SNS is guaranteed, since the majority of SNS offer open API that are based on the Web service paradigm. The possibility of giving a social texture to smart home applications is demonstrated in Section 8.1.

Chapter 6

Request Queues for Enhanced Performance of Smart Home Operations

Although wireless communications are becoming mature, transmission failures are a common happening, especially in indoor environments. These failures range from 1-5% in well-defined topologies, to even 20% in ad hoc scenarios where some embedded devices are mobile [40, 184]. Our observations indicate that typical smart home deployments with static topologies of devices experience around 3-5% of transmission failures.

Obviously, these failures degrade the performance of in-home operations, reducing the reliability of the system. Guarantees cannot be made regarding satisfaction of requests and thus high unpredictability is present. Nevertheless, transmission failures may be masked in part, by using intermediate request queues, which may be used for better managing the interactions with embedded devices. This is in concept similar to the TCP protocol in IP, which offers reliable transmission over an unreliable channel.

We define request queues as FIFO queues, installed on middleware applications for smart homes (e.g. our application framework) in order to handle the requests coming from the tenants, targeting the embedded devices of the house. Each device is associated with its own request queue,

while the FIFO property can offer fairness between tenants who concurrently invoke services offered by some particular device. Additionally, request queues enable multiple simultaneous requests, which are placed in the queue until they can be serviced, rather than being lost.

In this chapter, we examine the use of request queues as a suitable data structure for enhancing the performance of pervasive applications that target home automation. We consider request queues as a mechanism for supporting multiple home residents, who may concurrently interact with their smart space, as well as to use the queues for masking in part transmission failures, offering enhanced reliability and fault tolerance to the home environment. Moreover, request queues can also be designed with different service disciplines, e.g. load balancing and priorities, in accordance with application needs.

We need to stress the difference between intermediate request queues and queues that may be integrated on embedded devices. While the latter case could be used for offering support for requests from multiple home members, it would not provide any guarantees in case transmission failures occur. Moreover, various other benefits may be obtained by means of intermediate request queues. These benefits are discussed in Section 6.4.

We need to note that the use of request queues does not directly contribute towards the evolution of smart homes towards the Web. Nonetheless, it is crucial to address reliability and performance issues in smart home environments that host resource-constrained, embedded home devices. Besides, Section 4.2.3 shows that the synchronous operation at the Web can be combined seamlessly and smoothly with the asynchronous operation of the application framework and the request queue mechanism.

The rest of the chapter is organized as follows: Section 6.1 describes the experimental setup employed for configuring the request queue mechanism for good performance and for evaluating

potential applications that exploit the queing environment. Then, Section 6.2 analyzes theoretically the behavior and general functionality of the request queues, identifying potential advantages of employing queues. After, Section 6.3 investigates the setting of the retransmission interval to enhance the performance of the queues while Section 6.4 evaluates potential smart home applications that may be developed when request queues are used. Finally, Section 6.5 summarizes the findings of the analysis performed in this chapter, concerning the request queue mechanism.

6.1 Experimental Setup

This section explains the experimental setup used to properly adjust and configure the request queue mechanism, as well as to evaluate and test various applications of the queuing system.

6.1.1 Modeling User Behavior

In order to analyze the behavior of the request queues, realistic traffic from Web clients needs to be created. An emulated scenario was implemented, including multiple virtual Web clients, who interact with the sensor devices of their home space through the Web. The behavior of these clients was simulated by creating software agents who reside on the same computer as our application framework. Therefore, network delay is negligible. Using Restlet, we assigned a unique TCP socket to each agent. The operations of each agent include the *random* selection of a sensor mote, as well as the random invocation of a RESTful Web service offered by the device.

The arrival rate of Web clients at the framework is modeled by the exponential distribution. The exponential distribution is preferred to simulate real-world applications that deal with the occurrence of events, as for example the arrival of telephone calls or customers at a bank. The parameter λ denotes the arrival rate of the clients. For example, when $\lambda = 2$, this means that two clients arrive at the framework every second.

Given the nature of arriving customers, it is commonly accepted that the exponential distribution models the arrival of tenants' requests quite realistically and rationally. Furthermore, this is very desirable, as given the mathematical nature of the exponential distribution, a number of quite simple relationships can be derived for several performance measures based on knowing the arrival and service rate at each device.

6.1.2 Experimental Deployment in a Smart Home

During the analysis efforts, the application framework has been installed on a typical laptop (Intel Core Duo Dual Core 2.2 GHz), inside an experimental one-bedroom smart home. A wireless network of Telosb sensor motes was employed during the analysis, to sense the environmental conditions in a proof-of-concept smart home deployment. A sensor mote was plugged in one of the USB ports of the laptop to serve as a base station (see Section 2.1.3), for forwarding/receiving through the serial-over-USB port IEEE 802.15.4 wireless packets exchanged between the application framework and the remote Telosb motes.

Transmission power was set at -25 dBm, giving a transmission range of approximately 5 meters. The size of the 6LoWPAN messages was 128 bytes each. Each test has duration of 5 minutes. Telosb sensor devices were equipped with temperature, humidity and illumination sensors (see Section 5.1.1). Their sensing capabilities were exposed as RESTful Web services (see Section 5.2.3), transforming them into embedded Web servers.

We experimented mainly with a small number of motes (2-4 motes), deployed in a star topology inside the home environment, with 5 meters distance from the base station and from each other. This experimental setup is displayed in Figure 6.1. For demonstration purposes, in Section 6.4.1 we also considered a 3-layered WSN of sensor motes in a 2-3-4 topology.

We stress the fact that only a small number of devices was selected in order to test the request queue mechanism in an exigent case, in which high workload would exist targeting only a limited number of motes. Therefore, the request queue of each device would be effectively loaded with numerous consecutive requests. Increasing the number of motes serving a predefined workload results in distributing evenly the load between the devices, decongesting their request queues.

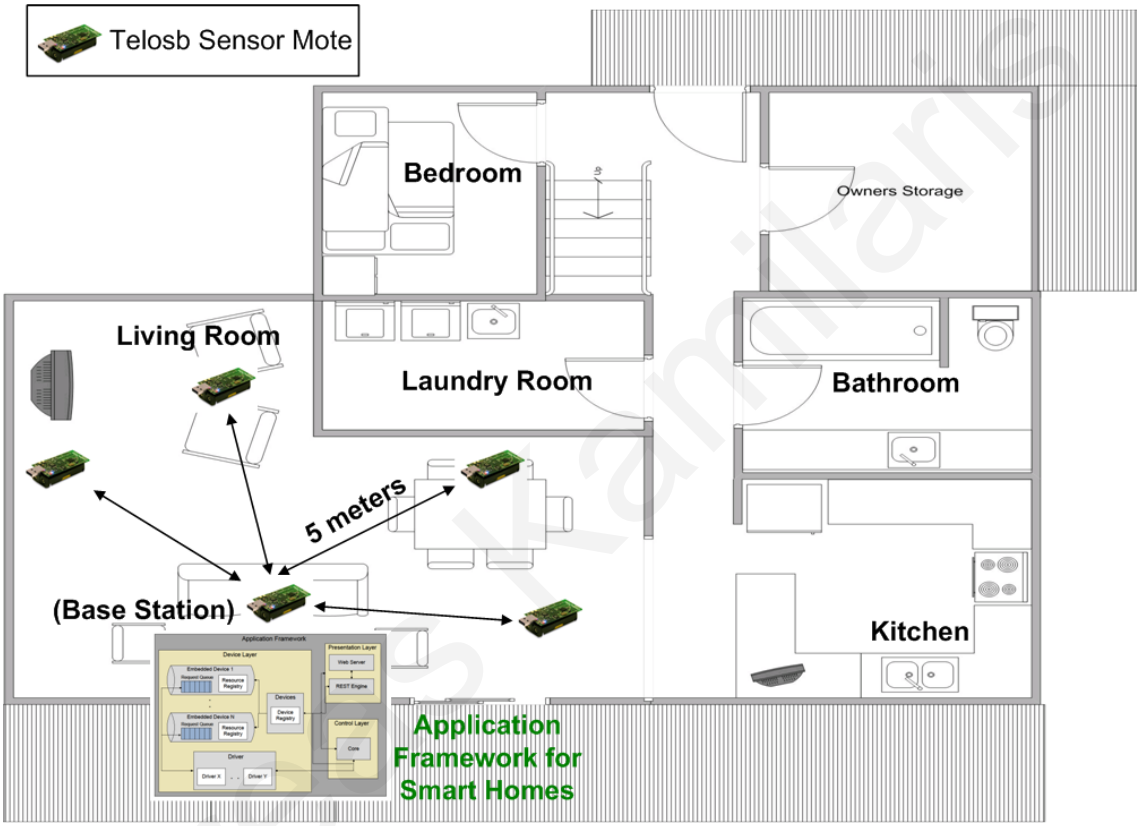

Figure 6.1: Experimental setup (single-hop star topology).

6.1.3 Parameters, Performance Metrics and Assumptions

Before proceeding to analyze the behavior of the request queue mechanism, we list here all the parameters, symbols, metrics and assumptions we include in our analysis procedure and experiments.

6.1.3.1 Parameters and Symbols

At first, the parameters and symbols we use are the following:

- **Arrival Rate** λ. The arrival rate of tenants' requests at the application framework, defined in requests per second.
- **Round Trip Time (RTT)**. The average amount of time from the wireless transmission of a request until the arrival of the response. This is actually the service time of the queue.
- **Request Queue Size** S_q. The current size of the request queue, i.e. the number of requests waiting their turn to be executed. This number does not take into account the request being currently executed.
- **Retransmission Interval** α. The fixed timeout a request queue waits for a response from an embedded device, for a previously made request. After this timeout interval elapses, the request is retransmitted to the device for a number of retransmission attempts. If no response is received after all attempts, then the device is considered unavailable.
- **Maximum Retransmission Attempts** n. The maximum number of retransmission attempts before a home device is defined as unavailable.
- **Probability of Transmission Failures** p_f. The probability of a transmission failure during request transmission from the application framework or a response transmission by the physical device.
- **Residual Service Time** R. This is the remaining time of the request being executed, if any. It is always less or equal the RTT time.

6.1.3.2 Performance Metrics

Furthermore, we list the performance metrics we use in order to assess the overall performance of the request queue mechanism:

- **Average Request-Response Time or Response Time**. The average amount of time needed from the creation of a request by a Web client to the arrival of the response, transmitted by a sensor device.
- **Waiting Time q**. The amount of time a request waits at the queue for its turn.
- **Scalability of home residents**. Effective and reliable communication with multiple home residents, interacting with their devices simultaneously through the application framework. Typical numbers of home residents are 5-10, however, workloads of up to 100 request-s/minute should be supported by the framework. This covers the case when each home resident uses different, personalized and intensive applications towards home automation and increased comfort.
- **Reliability**. Includes the reliable delivery of request messages from home residents to their home devices and vice-versa. Reliability involves no messages losses due to transmission failures and collisions.

6.1.3.3 Assumptions

Finally, a list of all the assumptions considered during analyzing the operation of the request queues is shown below:

- We assume that Telosb sensor motes and their services have already been discovered by the application framework. Thus, the minor delays associated with discovering the devices

and parsing their associated WADL files for understanding their functionality (see Sections 5.2.1 and 5.2.2), are not included in the experiments.

- Sensor devices are static during the tests without any mobility in the nearby area.
- The motes do not use any duty cycling for conserving energy, but they are always listening for incoming requests.
- Extra transmission failures are simulated randomly during the tests, based on the uniform distribution. Thus, subsequent transmission failures are not correlated. Since the sensor motes are placed in a predefined distance from the base station and from each other, the uniform distribution may simulate quite well any transmission failures that occur during the operation of the system.
- Possible interferences due to neighboring nodes (e.g. collisions in accessing the wireless medium) are not taken into account. We assume that any such collisions are random.

6.2 Analysis of the Request Queue Mechanism

The application framework associates each home device with a request queue, for supporting simultaneous tenants to send concurrently HTTP requests to them. The request queues have the responsibility of forwarding requests to the devices sequentially, according to the request time of arrival. As soon as a response arrives, the corresponding request is removed and the next request is forwarded to the head of the queue.

Figure 6.2 illustrates the operation of the request queue mechanism. Home tenants may pose simultaneously Web requests to some home embedded device with a request arrival rate λ. These requests enqueue at the request queue of the device and wait q time for their turn, in order to be transmitted to the physical device. Service rate is the average number of tenants served per second

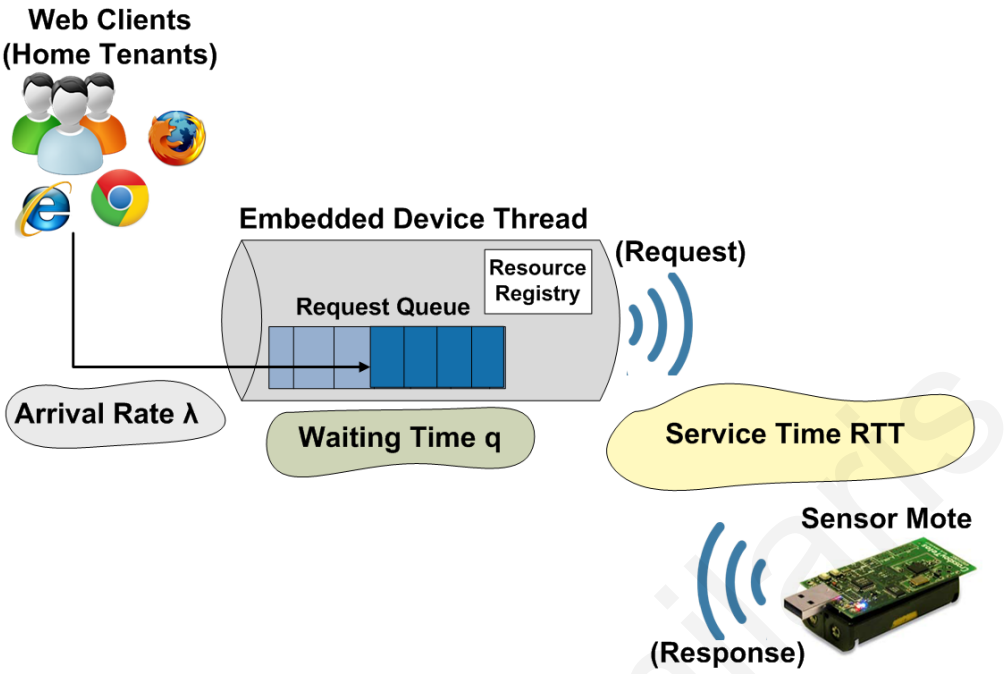

Figure 6.2: The intuitive behavior of the request queue mechanism.

by the device. The service time of the queue involves the time from which the request leaves the queue, it is satisfied by the sensor mote, and the response is received back at the device thread. We label this time as round-trip time (RTT).

Incoming requests need to wait in the queue for their turn, in order to be executed. Figure 6.3 shows a snapshot of the request queue load during a typical framework's operation, as recorded using the setup discussed in Section 6.1, for 3 different scenarios of arrival rates λ, of requests per second. These request rates can be considered high for typical smart home scenarios with few inhabitants, however, they are selected in order to stress test the system. As expected, as the arrival rate of requests increases, the size of the queue is increased as well. This size may be up to 4, when λ equals 1.5 and up to 7, when $\lambda = 2$. Obviously, the arrival rate λ of home residents affects significantly the request queue size. This can be observed in Figure 6.4, where the average size of the request queue of some sensor device is provided, for different values of λ. As the figure

shows, the size of the queue grows linearly with a small gradient in low traffic, until the arrival rate becomes 1.5. From this point, as the rate λ increases, the size of the queue grows with a significantly larger gradient. This is a possible sign of overloading the queue (see Section 6.2.2), and is as expected in theory.

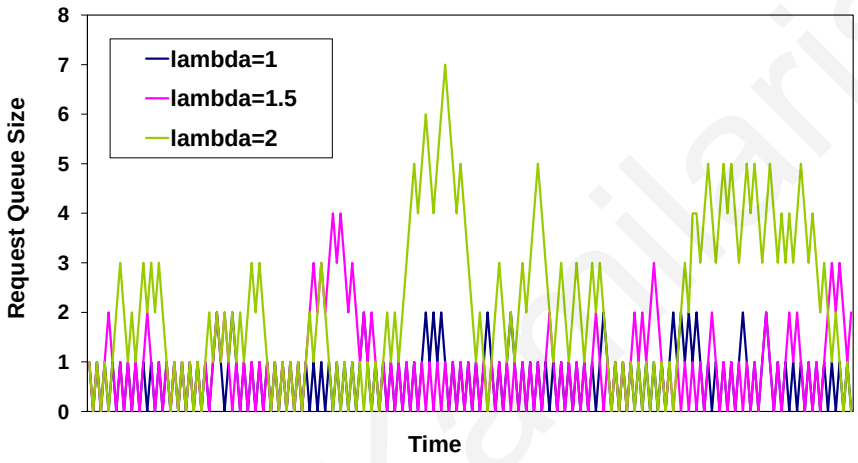

Figure 6.3: Load at the request queue of some sensor device during its operation.

Of course, a large size of the request queue implies a larger waiting time for incoming residents' requests. This waiting time at the queue can be a significant source of delay, as depicted in Figure 6.5 for the case when $\lambda = 2.5$, in a heavily loaded scenario. As the different measurements show, requests need to wait a few seconds when the size of the queue is more than 2. Hence, reducing the waiting time of requests is expected to have a major impact on the system's performance and on the satisfaction of the home tenants.

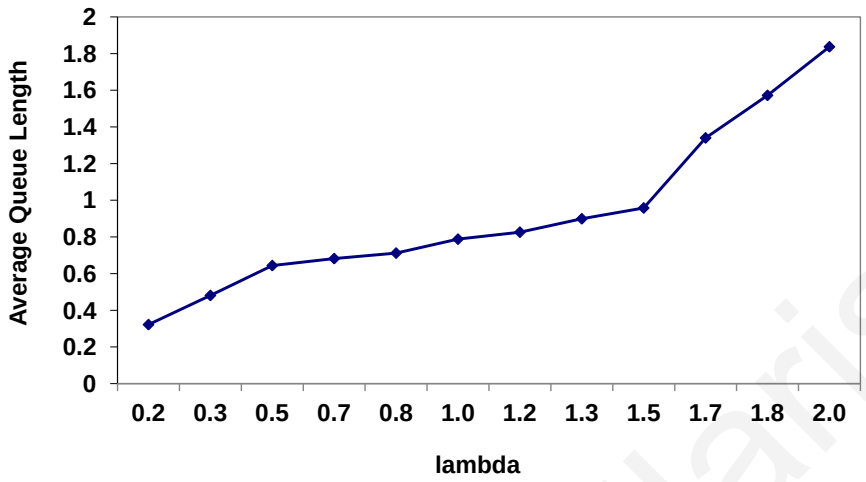

Figure 6.4: Average size of the request queue of some sensor device for different arrival rates.

Moreover, during the framework's operation, in case some transmitted request/response message is lost, the queue waits some amount of time and then it retransmits the message. This retransmission feature aims at enhancing the reliability and robustness of the application framework. However, this comes at the expense of additional delay, which is influenced by the design parameters, as we discuss below. A comprehensive state diagram of the request queue mechanism, including the retransmission feature, is shown in Figure 6.6. In the figure, all the states of the queue, as well as possible events/actions and transitions are provided.

In the following subsections, we perform some basic theoretical analysis of the request queue mechanism, identifying some potential advantages of adopting a queuing-based model for managing the interaction with home devices. For example, we may estimate the response time needed

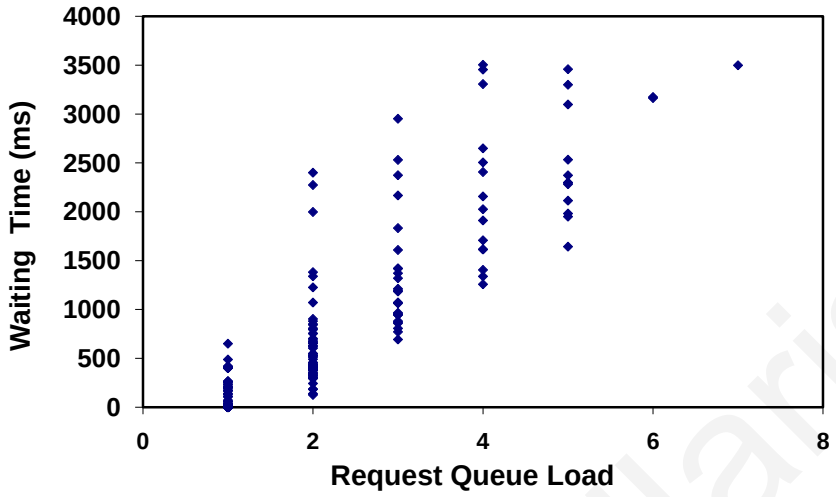

Figure 6.5: Waiting times of the requests at the request queue of some sensor device for $\lambda = 2.5$.

at some time in order to satisfy a request or monitor whether the request queues are becoming overloaded.

6.2.1 Estimating Potential Response Times

An important aspect is the capability of predicting the average response time, needed for satisfying some request. Obviously, in the case of no transmission failures:

$$Average\ Response\ Time = q + RTT \tag{1}$$

In case transmission failures occur, average response time increases according to the equation:

$$Average\ Response\ Time = q + RTT + \alpha \cdot p_f \cdot (\frac{1 - p_f^{n-1}}{1 - p_f}) \tag{2}$$

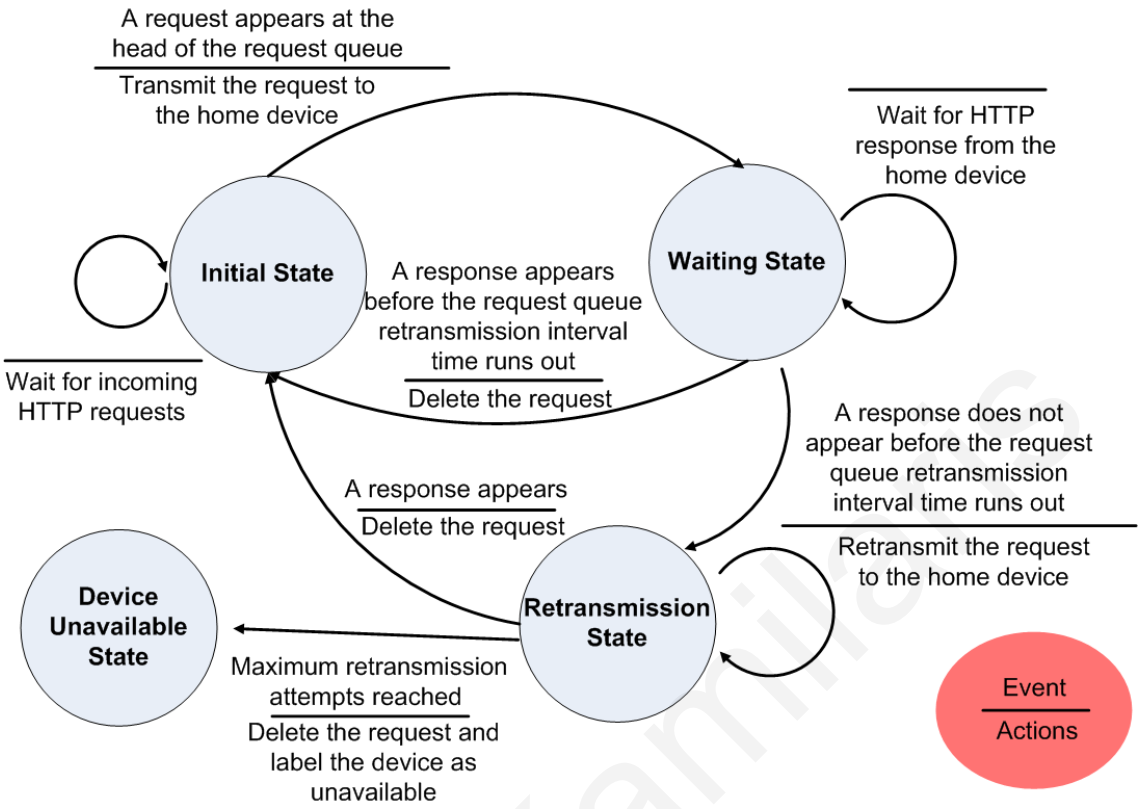

Figure 6.6: A simplistic state diagram of the request queue mechanism.

Considering that the waiting time q of some specific request equals the residual time R of the request being executed and the response times of all other S_q requests waiting already at the queue, we can define q as:

$$q = S_q \cdot RTT + R + (S_q + 1) \cdot \alpha \cdot p_f \cdot (\frac{1 - p_f^{n-1}}{1 - p_f}) \tag{3}$$

We take into account that any request waiting at the queue has a certain probability of transmission failure(s), including the current request at the head of the queue being executed. Hence, the average response time may be calculated according to the following equation:

$$Average\ Response\ Time = (S_q + 1) \cdot RTT + R + (S_q + 2) \cdot \alpha \cdot p_f \cdot (\frac{1 - p_f^{n-1}}{1 - p_f})) \quad (4)$$

R is estimated by recording the dequeue time of the request being executed and subtracting this time from the average RTT time. Equation 4 will be evaluated experimentally in Section 6.4.3.

6.2.2 Avoiding Overloading the Request Queues

The stability condition of each request queue may be roughly estimated by observing the arrival rate of incoming requests λ. To achieve stability, it must be ensured that service time is less than the inter-arrival time, i.e. $RTT < \frac{1}{\lambda}$. In our experiments in Section 6.3.1 (with $RTT = 530\ ms$), this implies that $\lambda_c = 1.88$ *for each* sensor device, i.e. the arrival rate λ must be always smaller that λ_c to avoid overloading the queues.

In case λ exceeds the critical value λ_c, a "harsh" strategy would be to start dropping requests, e.g. last arriving requests at the queue, or random requests already in the queue, or even adopt other measures. Some initial discussion of a number of measures, such as load balancing and support for prioritized requests appear in later sections (see Sections 6.4.4 and 6.4.5). However, a comprehensive strategy is beyond the scope of this thesis.

In theory, this may be generalized for n devices, assuming that requests are distributed uniformly to them. In this case, the total arrival rate λ may reach $n \cdot \lambda_c$ requests per second. However, in practice, such large arrival rates cannot be supported because of other factors affecting the system's behavior such as the framework's processing delay and the transmission throughput of the sensor mote acting as the proxy between the Web and the physical environment (base station).

In a real-world setting, the stability condition might be more appropriate to be averaged over a relatively large period of time, e.g. every 30 minutes or some hours, to avoid dropping requests in unexpected rush hours that last only some minutes.

6.2.3 Considering other Measures of Interest

By observing the operation of the queues and the service capabilities of home devices, various measures of interest may be calculated. For example, the utilization of the device is the arrival rate at the device multiplied by the average RTT time. The expected number of clients equals the average response time multiplied by the arrival rate at the home device (Little's law).

While such calculations are based on observations and experience, more theoretically accurate results could be obtained by applying principles from queuing theory. For example, by observing the RTT times of some sensor mote, a histogram of the response times may be obtained. This histogram could indicate the service rate of the device, which may follow any general statistical distribution. Assuming that the arrival rate of tenants is derived from the exponential distribution (see Section 6.1.1), using Kendall's notation, the queuing model of this particular device may be defined as a M/G/1 queue [34].

Hence, by applying Pollaczek-Khinchin (PK) formula, the expected number of Web clients using the system can be inferred. Then, an application of Little's theorem provides the average time spent in the queue by the clients. More advanced, queuing theory could define the average expected waiting times, the distribution of the length of the busy periods, the distribution of the sojourn and waiting times, the distribution of the number of arrivals during the service time etc. Nonetheless, this theoretic analysis is left for future work.

6.3 Configuring Request Queues for Enhanced Performance

The amount of time from a transmission loss of a request/response message until its retransmission is crucial for the system's performance, since fast retransmissions would increase the service rate of the requests. However, retransmissions that happen too early may encounter the

likely event that the original request message had already been served, but the response has not arrived back yet. Such premature timeouts could cause unnecessary additional loading that increases collisions in a loaded system, thus leading to system inefficiencies.

Hence, fine-tuning the request queue retransmission interval is important for the framework's operation. In the following subsections, the main focus is to analyze the queue behavior, and properly set the value of the retransmission interval. We refer to the request queue retransmission interval parameter as α.

6.3.1 Searching for an Effective Retransmission Interval

The average RTT of each sensor mote participating in the experimental setup may be derived through sampling. Figure 6.7 presents the histogram of some sensor node deployed in one-hop distance from the base station. In this case, the average RTT time in the one-hop communication with the motes is 530 ms with a standard deviation around 70. Having this in mind, if $\alpha < RTT$, numerous useless retransmissions would happen, deteriorating the overall system performance. This can be observed in Figure 6.8. Hence, we only examine values of $\alpha \geq RTT$. However, since useless retransmissions still exist, even with slghtly larger values of α, we increase α until retransmissions are effectively reduced near to zero. This happens when α is about 600 ms.

We need to note that when α is less than 600 ms, the system does not operate correctly. The percentage of (unnecessary) retransmission attempts in relation to the total number of client requests grows exponentially as α decreases since the queue incorrectly believes that the request messages keep failing. In the special case when $\alpha = RTT$, false retransmission attempts reach 18%, or in total 54 redundant retransmissions when $\lambda = 1$ in 5 minutes simulation time. Continuous retransmissions cause the sensor devices to malfunction, reduce their battery lifetime and increase the overall loading of the system.

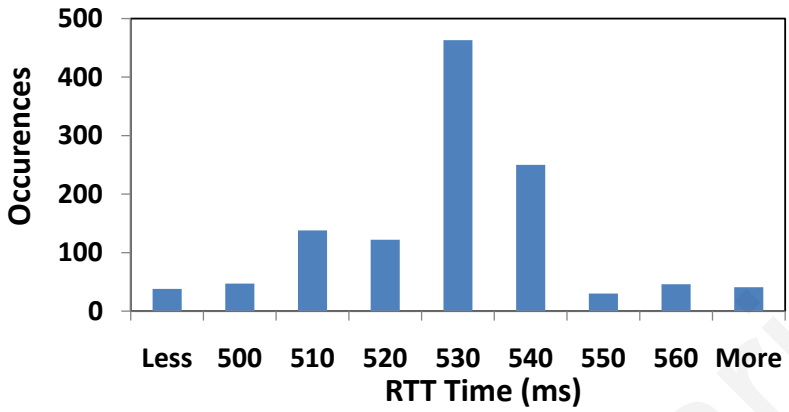

Figure 6.7: Histogram of RTT times of some sensor mote in one-hop distance.

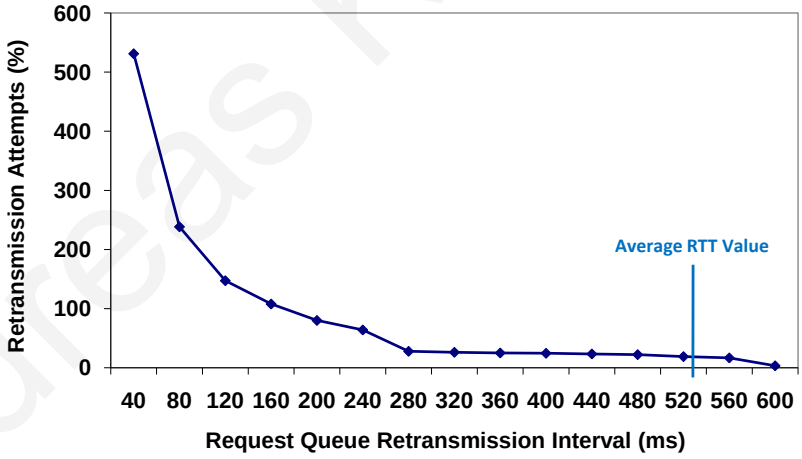

Figure 6.8: Retransmission attempts in low values of the request queue retransmission interval.

The experimental setup for investigating the parameters affecting the retransmission interval of the request queue involved two sensor devices. A small number of devices was selected, as the aim was to observe the queues when having considerable load. The first experiment studies the effect of transmission failures on the request queues, using $\lambda = 1$. By default, the transmission failures in our deployment are around 3-5%, so we emulated additional failures by manually dropping packets at the driver module. Figure 6.9 presents the response times in relation to the retransmission interval α, considering different percentages of failed transmissions.

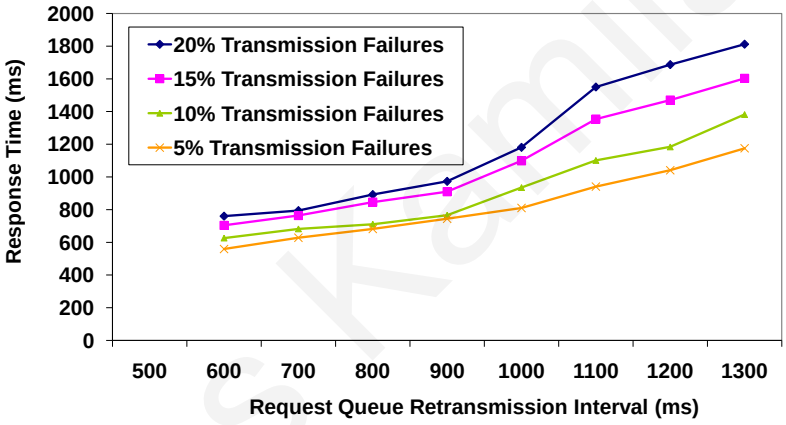

Figure 6.9: The effect of transmission failures on the request queues.

As transmission failures increase, response times are also increased, but not at an alarming rate. In all scenarios of different percentages of message failures, $\alpha = 600\ ms$ yields the best response times. As the parameter α increases, response times are increased as well. An important thing to notice is that as the percentage of transmission failures increases, the slopes of the graphs become larger. This fact denotes that the request queue mechanism becomes more important

when transmission failures exist in some home environment, since the proper adjustment of the retransmission interval parameter affects more significantly the response times.

The next experiment examines the retransmission interval α in varied workload. In this and the next experiments, the default case of 3-5% transmission failures is used. This is realistic for typical smart building environments. The results of this experiment are displayed in Figure 6.10.

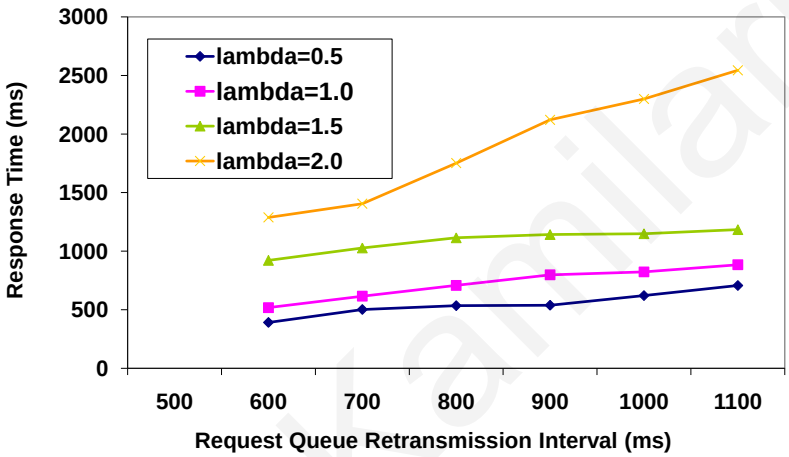

Figure 6.10: The effect of varied workload on the request queues.

Low traffic implies a small percentage of retransmissions. Thus, the influence of the request queue in the overall system performance is small. However, as the traffic increases, the retransmission interval α becomes more important, especially when $\lambda = 2$. Also in this experiment, as expected, the most appropriate value for α is 600 ms.

6.3.2 An Equation for Setting the Retransmission Interval

Considering that the average RTT time in the experimental setup is 530 ms with a standard deviation equal to 70 (see Figure 6.7), we assert that an appropriate value of α equals the RTT time plus its standard deviation value.

To examine whether this assumption is valid, we performed a last experiment, investigating the influence of RTT time on the request queue, as shown in Figure 6.11 for $\lambda = 1$. To increase RTT time, we programmed the TinyOS application which was installed on the sensor devices to wait an extra delay before transmitting a response back to the framework.

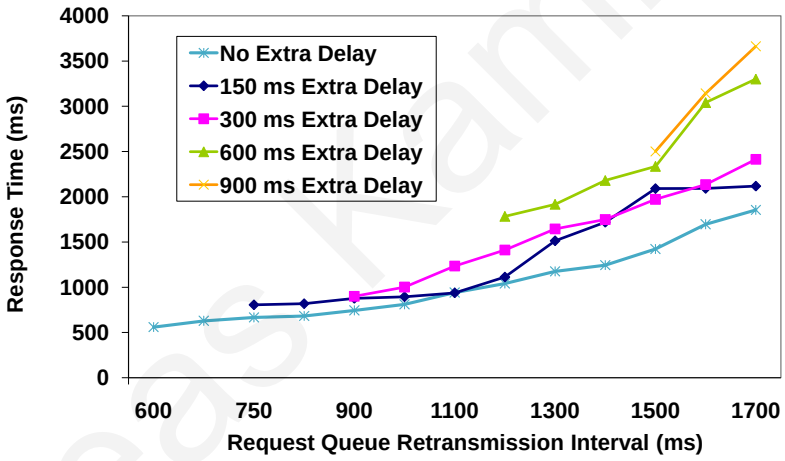

Figure 6.11: The effect of round trip time on the request queue.

As the RTT delay increases, a good choice for the value of α remains the RTT time plus some ”safety margin”. This safety margin can be related to the standard deviation. Therefore, we recommend that the α value should be set according to the following equation:

$$\alpha = RTT + c \cdot St.\ Deviation \tag{5}$$

where the parameter c allows the system to choose being more or less cautious. In a static deployment of home devices, similar to our emulated experimental setup, c could be equal to 1. In a dynamic case with mobile devices and changing topologies, c could take larger values. We discuss about this scenario in Section 6.3.3.

Of course, RTT times and standard deviation values are learned from the device thread after observing the physical device for some amount of time. Therefore, the best approach is to set initially α to a larger value, leaving a safety margin to avoid producing continuous retransmissions. During the device operation, the value of α may be fine-tuned accordingly, by considering the observed RTT times.

6.3.3 Support for Moving Objects and Dynamic Environments

The experimental setup described in Section 6.1 focuses on a static deployment of embedded devices inside a smart home. Hence, the average RTT times of the sensor motes were stable and constant. However, in a dynamic scenario with mobile devices and fast changing topologies, highly varying RTT times could be observed.

In this case, to estimate RTT, one could adopt exponential weighting, the approach used at the TCP retransmission timer mechanism [170]. Both a dynamic topology of devices in an indoor environment and the Internet are characterized by unpredictability and dynamic network changes. Therefore, RTT could be set according to the TCP equation:

$$RTT_{est} = (1 - \beta) \cdot RTT_{est} + \beta \cdot RTT_{sample} \tag{6}$$

where RTT_{est} is a smoothed RTT time, influenced by past RTT samples, RTT_{sample} is the current measurement of RTT and $0 \leq \beta \leq 1$.

Similarly, standard deviation could be defined according to the TCP equation:

$$RTT_{dev} = (1 - \gamma) \cdot RTT_{dev} + \gamma \cdot |RTT_{sample} - RTT_{est}| \quad (7)$$

where RTT_{dev} represents RTT variation or approximately the standard deviation and $0 \leq \gamma \leq 1$.

The parameters β and γ define the memory capability of the system, to remember the history of measurements and slowly adapt to changes in RTT and standard deviation values. For fairly stable environments, these values could be set close to 0, while in more dynamic environments closer to 1.0. Obviously, if more smoothing is required, e.g. in case RTT changes frequently, lower values of β and γ may be selected. In TCP, a typical value of β is 0.125 and of γ is 0.25.

Finally, the parameter c in Equation 5 was set equal to 1 in the static case. However, in a dynamic scenario with mobile devices and fast changing topologies, c should take larger values, to adapt more easily to the possibly highly varying RTT times of the embedded devices. Nonetheless, experimenting with these parameters to adapt to dynamic environments is beyond the scope of this thesis, due to the highly varying characteristics of different application scenarios.

6.4 Potential Benefits of Using Request Queues

By using request queues at the application framework for managing the communication with embedded devices, numerous benefits can be obtained. For example, request queues could be used for offering load balancing between embedded devices, for supporting prioritized requests and for providing some indications about the execution time of future requests, based on the queue load and status. We list and investigate some of these benefits in the following subsections.

6.4.1 Multi-Client Support

By employing request queues, multiple tenants may interact simultaneously with their home environment. Even though their requests could be delayed in increased traffic conditions, it is

guaranteed that they would eventually be satisfied. The average response times in varied workload are depicted in Figure 6.12, both for a single-hop and a 3-hop topology. The single-hop case involves 4 sensor motes in a star topology while the 3-hop case involves a 3-layered 2-3-4 topology (see Chapter 7, Figure 7.1), querying only the 4 leaf nodes for their services.

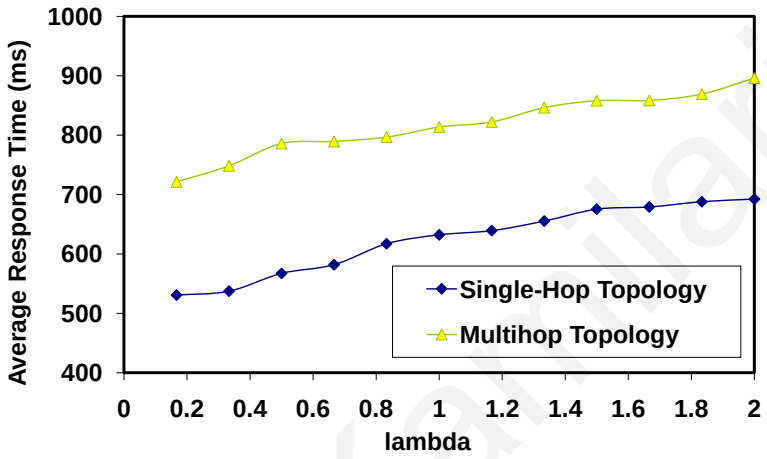

Figure 6.12: Average response times in different traffic conditions.

As the figure denotes, the multi-hop topology causes additional delays of around 200 ms to satisfy the requests. Heavy workload increases response times by 18-20% in the single-hop case and 14-17% in the multi-hop topology, while the increase is almost linear. This is an indication that response times scale well as the traffic increases. This fact suggests that hundreds of tenants may be supported by the application framework with acceptable performance.

6.4.2 Avoiding Transmission Failures

As mentioned earlier, transmission failures are successfully masked by means of the request queues. As soon as a failure is ”suspected” by the queue (e.g. a lost request/response message or a delayed response is sensed by the retransmission timer), the request is immediately retransmitted to the device. Hence, the system becomes more reliable and robust, with a small penalty on response times due to the retransmission attempts.

Figure 6.13 displays the response times in different percentages of transmission failures. In light workload, transmission failures do not affect significantly the response times. However, in heavy workloads, transmission failures cause the response times to grow almost exponentially. Nonetheless, even in the extreme scenario of 30% of failures, all requests are eventually satisfied, in near double time than in the 0% case.

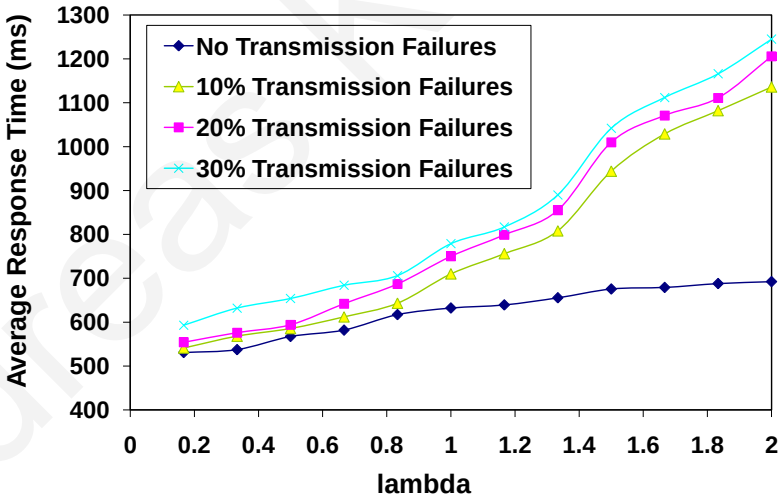

Figure 6.13: Average response times in different percentages of transmission failures.

6.4.3 Estimation of Response Times

In Section 6.2.1, we discussed theoretically how response times may be estimated when the request queue is at a certain state. This estimation is enabled by Equation 4. Here we investigate experimentally whether this equation may be adopted for this purpose.

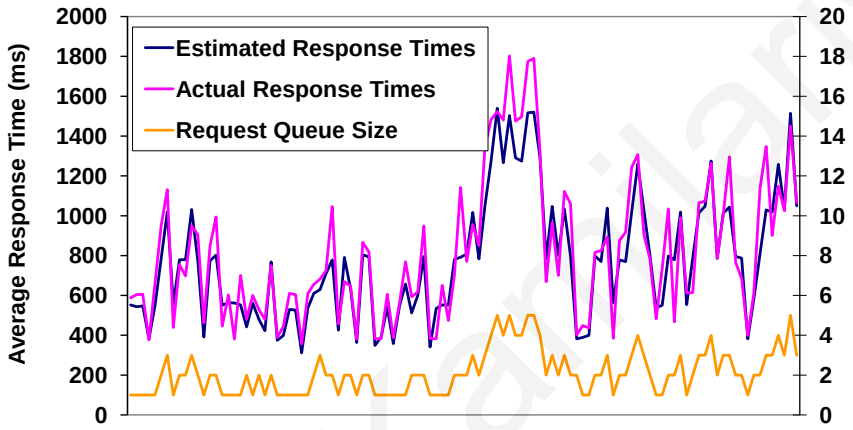

Figure 6.14: Estimation of response times Vs Actual response times.

The estimated response times, as well as the actual response times for some execution of the emulation are provided in Figure 6.14. In this scenario, only one sensor device was used with an arrival rate of residents equal to 2.5, in order to overload the request queue of the device and take a snapshot of its operation in high traffic. The default case of 3-5% transmission failures was adopted. As the figure shows, the estimation of response times behaves very well, with the model following the trends fairly accurately, except in cases when transmission failures happen. In this case, the response times are underestimated. Smaller errors are due to RTT deviations. In general,

the average estimation error is 12.38% when the request queue size is up to 2, and it increases to 14.60% when the request queue size becomes up to 6, since it is more possible for transmission failures to occur.

6.4.4 Load Balancing for Serving High Traffic

Another possibility to effectively serve increasing traffic would be to employ multiple (redundant) sensor devices. An equivalent example from real life concerns a queue at a bank, which employs a number of employees, in order to serve efficiently its customers. In this case, a module acting as a load balancer may be considered for managing a network of queues. This module could be placed on the application framework between the driver module and the device threads. To demonstrate this capability of using request queues, we implemented a load balancer using Algorithm 1.

Algorithm 1 A simple algorithm for load balancing.

1: Whenever a new request appears for service s, estimate the potential response time (see Section 6.2.1) for every device that offers the service s.

2: Forward the request to the embedded device that offers the least estimated response time.

A naive approach would be to select the device with the currently smallest queue size. However, each device type has a different service rate, while even devices of the same type may be in different hop-distance from the base station (varying RTT times). Hence, it is better to consider request forwarding based on Equation 4.

The results of the execution of this simple load balancing mechanism in varied workload can be observed in Figure 6.15, by employing 4 sensor motes in a star topology. In low traffic, the load balancer does not make much difference, since the requests are effectively served by the devices, even when they are randomly selected. There exists a small improvement of performance, around

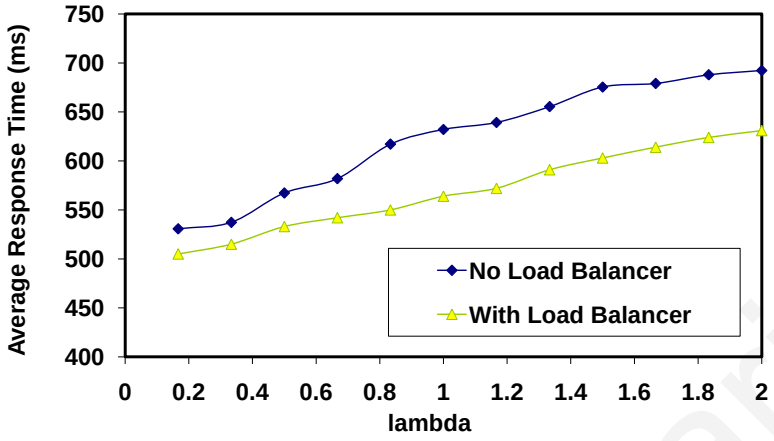

Figure 6.15: Waiting times for requests when using a load balancer.

4-6%. However, in increased traffic, the load balancer forces the incoming requests to be balanced evenly among the available devices. Thus, waiting times at the queues are decreased and the overall response times are effectively decreased. This decrease reaches almost 11%.

Load balancing may also increase the battery lifetime of the embedded devices, because the requests are distributed uniformly among them, In this way, the overloading of some device with requests is avoided and its battery is maintained for a larger time period.

6.4.5 Handling Priorities

The use of request queues may support prioritized requests without much effort. Priorities are considered in cases when an order of importance or urgency needs to be maintained. Typical categories of priorities may be *low*, *normal* and *high*. High-priority requests are the most important, while low-priority requests the least significant.

In smart home applications, using prioritized requests may be beneficial when some urgent operation needs to be performed as fast as possible. This operation may involve triggering the alarm when a thief breaks into the building or switching off an electric appliance that consumes excessive amounts of energy.

The request queues may be easily transformed into priority queues by converting the queues into priority heaps. A key characteristic of binary heaps is that they keep at the head of their binary tree the request that has the highest priority. The algorithm we used for defining priorities in the home ecosystem is described in Algorithm 2.

Algorithm 2 An algorithm for supporting priorities.

1: Assign a priority to each request (low, normal or high).

2: Each prioritized request is translated into an integer value by the application framework, according to the following formula: low=1, normal=5 and high=10.

3: The priority heap selects for execution the request with the highest priority number.

4: To avoid starvation of low-priority requests, the priorities of all waiting requests are increased by 1 at every round, i.e. at a successful execution of some request.

To test the priority queuing mechanism, the emulation described in Section 6.1 was employed with 2 motes. Figure 6.16 shows the response times of the prioritized requests according to the current size of the heap when each request appeared. This emulation was performed by setting $\lambda = 3$, including 10% high-priority requests, 10% low-priority and the rest with normal priority.

According to the figure, high-priority requests are executed first, with waiting times less than 2 seconds even when the heap has a size equal to 9. On the contrary, low-priority requests need much more time to be executed, reaching 10 seconds in some cases. Table 6.1 shows the load conditions of the priority heap at each priority category. As expected, high-priority requests remain the least

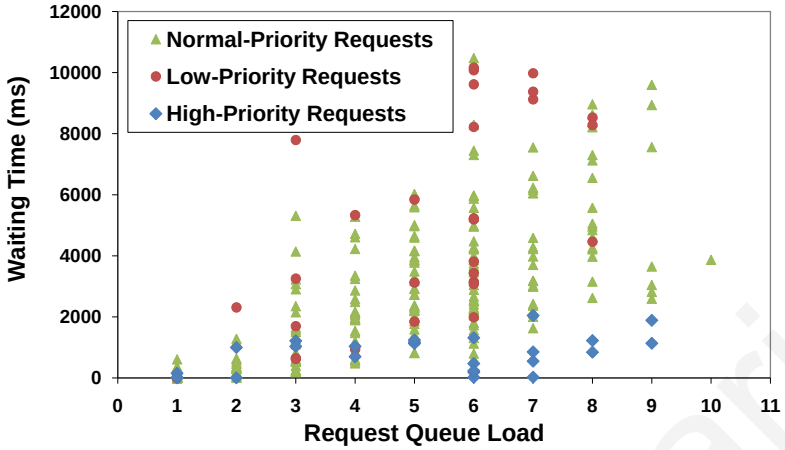

Figure 6.16: Waiting times for requests with different priorities and priority heap conditions.

time in the heap, between 1-3 rounds. Low-priority requests stay in average 5 rounds in the heap, while normal-priority requests fluctuate between low- and high-priority requests.

Table 6.1: Priority heap load conditions at the different priority categories.

Priority	**Min. Heap Load**	**Max. Heap Load**	**Avg. Heap Load**	**St. Deviation**
Low	1	12	5.40	3.808
Normal	1	9	3.24	2.421
High	1	3	1.44	1.003

The average waiting times of the priority categories when changing the percentage of high-priority requests are presented in Figure 6.17. As the number of high-priority requests increases, their waiting time is likewise slightly increased. This happens because a few high-priority requests might enter the queue concurrently. In such cases, a time priority is followed and the high-priority requests appearing later need to wait those appearing earlier. As the percentage of high-priority

requests increases, also the waiting times of the low- and normal-priority requests are increased. This occurs since the high-priority requests are executed first, delaying the execution of lower-priority requests.

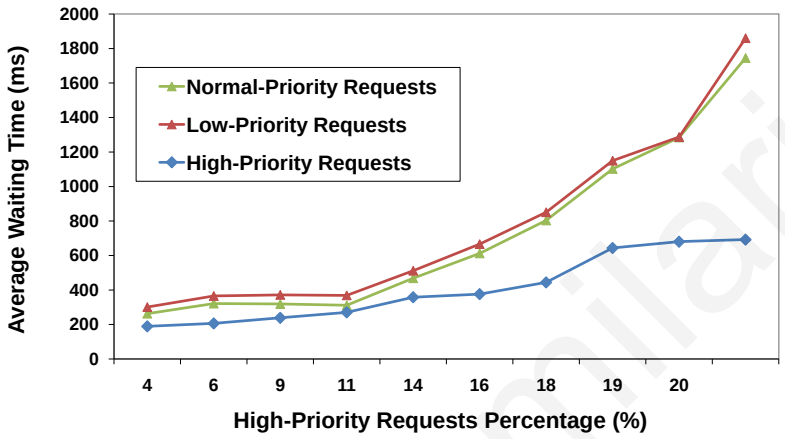

Figure 6.17: Average waiting times for different percentages of high-priority requests.

Figure 6.17 indicates that the waiting time of high-priority requests scales well, as their percentage increases. This happens by forcing lower-priority requests to stay longer in the queue, with an exponential growth of their waiting time, as high-priority requests are increased. Algorithm 2 may be easily extended to support more layers of priorities.

Offering support for priorities is not a native feature of the Web, even though it may be supported in the future. Nevertheless, priorities constitute an important feature in smart home applications that support multiple users and experience high traffic. They can be employed internally by the application framework, e.g. to favor the requests coming from specific family members or

those marked by the users as more important in some rule-based smart home application, similar to the one in Section 4.3.4.

6.5 Summary of the Request Queue Mechanism Analysis

This chapter examined request queues as a data structure for handling the interactions with embedded devices in a smart home. An analysis of their functionality was performed and potential advantages were identified and discussed. Properly setting the request queue retransmission interval is important for the efficient operation of the queuing system. Request queues may constitute a promising mechanism for enhancing smart home applications with numerous benefits.

Generally, request queues become more useful in scenarios with multiple, concurrent users and an increased percentage of transmission failures at the home space. They may be well used in scenarios that demand support of prioritized requests and they can serve high traffic through load balancing and device redundancy. Request queues can be considered as a dynamic, adaptive system that handles failures in the home environment providing certain reliability.

Furthermore, we identified the similarity between the retransmission interval time and the retransmission mechanism used at the TCP protocol. This fact encourages further research that concerns applying Internet principles in smart home applications for added performance, including probably moving objects and more dynamic environments.

Finally, the use of queues enables the detailed analysis of the system by employing queuing theory. Queuing theory could answer more complicated questions and be exploited to provide some guarantees and QoS indications to home tenants.

Chapter 7

Evaluating Web Techniques for Enhancing Smart Home Performance

In this chapter, a technical evaluation of an IPv6-based solution for Web-enabled smart homes is performed. This evaluation focuses mainly on Telosb sensor motes (see Section 5.1.1), as they are expected to accept heavy workloads due to their continuous monitoring operation. The performance of sensor motes reflects the operation of any Web-enabled embedded device, including our Ploggs smart power outlets (see Section 5.1.2).

In general, smart meters, smart power outlets and smart appliances do not demand high performance, as their primary goal is to inform the system in pre-defined, relatively slow intervals about the energy consumption of electrical appliances or the total consumption of the smart home. Smart power outlets and smart appliances are used also for controlling individual appliances, but this operation is expected to happen sporadically, for example when some energy-efficient smart rule is validated through the physical mashups editor (see Section 4.3.4) or when the smart home needs to urgently switch off some electrical appliances in a load shedding scenario (see Section 9.3). Nevertheless, in Section 9.2 we provide an evaluation of Ploggs in terms of response times, in a case where the smart home exploits the demand response program of the smart grid to schedule tasks in low-tariff hours.

Our technical evaluation spans three different scenarios. In the first, request/response interaction is explored, following an emulation where a large family interacts with the smart home through the Web. An IPv6-enabled WSN is deployed in an experimental smart home, considering both a single-hop star topology and a tree topology with multi-hop communication between sensor motes. In the second, the energy consumption of sensor motes is investigated, by measuring the energy performance of the 6LoWPAN-enabled WSN both in request-response and event-based messaging. Finally, the third scenario examines streaming of sensor measurements, where events are continuously triggered with high frequency and need to be forwarded (pushed), from the application framework (publisher) to third-party applications (subscribers).

By evaluating request-response, event-based and streaming messaging, all the three interaction possibilities with home devices are met, as specified in Section 4.1, where the requirements for smart homes are defined. These three interaction patterns cover the most important and popular use cases that shall be supported in future smart home applications.

The rest of the chapter is organized as follows: Section 7.1 describes the experimental setup during the evaluation procedure and the performance metrics used. Then, Section 7.2 examines the response times of the sensor motes in a request-response interaction model. Afterwards, Section 7.3 considers the energy performance of the sensor motes, both in a request-response and a event-based messaging scenario and, finally, Section 7.4 evaluates the behavior of the system in a streaming case, where streaming events need to be forwarded to interested home tenants.

7.1 Experimental Setup

In the following evaluation scenarios, both a single-hop and a multi-hop topology are examined and compared. The single-hop case is similar to the experimental setup used during the

analysis of the request queue mechanism (see Section 6.1). It involves a star topology inside an experimental home environment and it can be graphically observed in Figure 6.1.

On the other hand, the multi-hop case concerns a tree topology with 3 layers, in a 2-3-4 formation of sensor devices. In this case, sensor motes have 5 meters distance from their parent node and from each other. This topology is depicted in Figure 7.1. We stress the fact that only the 4 leaf nodes, located in a 3-hop distance from the base station are used for satisfying requests. This happens to stress the operation of the system, taking into account the extra multi-hop delay.

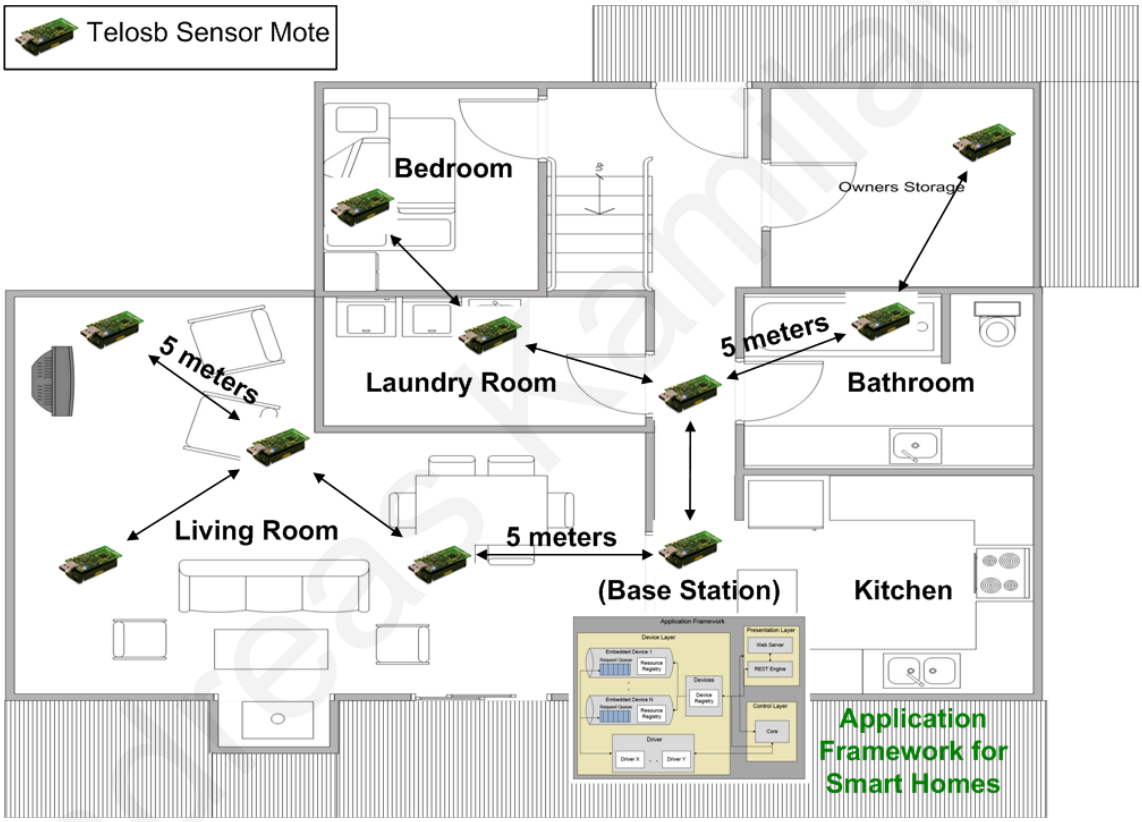

Figure 7.1: Experimental setup (multi-hop).

Similar to the experimental setup in Section 6.1, realistic traffic from Web clients was emulated by programming multiple virtual Web clients interacting with the sensor devices of their smart home through the Web. These clients are modeled as software agents residing on the same laptop

as the application framework. Thus, network delay is negligible. Their arrival rate λ follows the exponential distribution. Each test has duration of 5 minutes. All the assumptions considered during this technical evaluation are listed in Section 6.1.3.3.

We note that although our emulation assumes multiple residents making continuous requests to their smart home devices, in real life these requests could come from agents of the residents, residing in some remote location(s) and interacting with the home environment for monitoring purposes. In this case, a M2M communication would be enabled, with each resident having one or more agents guarding the house for possible abnormalities.

7.1.1 6LoWPAN Vs Simulated REST

For comparison reasons, we also programmed our Telosb sensor motes without 6LoWPAN support, using the native programming model of TinyOS, still following the REST architectural style. Thus, we developed *simulated REST*, which is a light, simulated version of REST without the IPv6 overhead. In simulated REST, the device discovery procedure employed is simplistic and not based on any Web standards. Obviously, simulated REST is more efficient, because it uses the native messaging stack of TinyOS without the IPv6 burden. From now on, the 6LoWPAN-based WSN is referred to as *6LoWPAN REST* and the version without the IPv6 support as *Simulated REST*. In our evaluation efforts in the following sections, we measure the performance also for simulated REST, in order to develop effective and reliable comparisons.

The details of the memory usage of Telosb sensor motes are given in Table 7.1. The size of the 6LoWPAN messages is 128 bytes while the size of messages in simulated REST is 80 bytes. The memory footprint of the 6LoWPAN REST software is larger than the simulated REST code, due to the additional libraries of Blip.

Table 7.1: Memory usage of Telosb sensor motes.

Module	**Memory Footprint**
6LoWPAN REST TinyOS Software	40,604 Bytes
Simulated REST TinyOS Software	29,860 Bytes
6LoWPAN REST Message Size	128 Bytes
Simulated REST Message Size	80 Bytes

7.1.2 Performance Metrics

In the following tests, we focus on the following performance metrics:

- **Average Request-Response Time or Response Time**. The average amount of time needed from the creation of a request by a Web client to the arrival of the response, transmitted by a sensor device.

- **Push Time**. The average amount of time needed to forward events from the application framework to interested third parties.

- **Cache Hits**. The serving of a request through the Web caching mechanism (see Section 4.2.5), instead of transmitting the request to the physical home device, through the wireless medium of the home environment.

- **Energy Consumption**. The energy consumed by the home devices during their operation in Joules.

- **Scalability of home residents**. Support for multiple, simultaneous home residents (and requests) by the application framework. These residents may interact concurrently with their

home devices. Although typical families consist of 5-10 members, the application framework should support also larger numbers of users/requests, e.g. up to 100 requests/minute. This would cover the case that some home tenants employ Web agents that monitor the house and take informed decisions for home automation.

- **Scalability in terms of topologies of home devices**. Effective and reliable communication both in a single-hop star topology as well as in a multi-hop tree topology in the home wireless network.

- **Reliability**. It is about the reliable delivery of request messages from home residents to their home devices and vice-versa. Reliability involves no messages losses due to transmission failures and collisions.

7.2 Response Times in Request-Response Interaction

In this first experiment, the performance of the request-response interaction in the smart home is evaluated, in terms of response times. As mentioned before, 2 different topologies are considered to test the system, namely a single-hop star topology and a multi-hop tree topology. In both, the existence of a large family is assumed, interacting with the smart home (see Section 7.1).

Multiple virtual residents are interacting with the IPv6/6LoWPAN-enabled sensor motes of their smart home through the Web. Selection of sensor motes and RESTful Web services is performed *randomly* by the residents. Various tests are performed with variable numbers of residents, ranging from 10 to 120. Assuming that each resident performs 1 request per minute, residents are modeled by varying the arrival rate λ. In this modeling, an arrival rate of $\lambda = 1$ means that 60 residents interact with their house every minute.

7.2.1 A Single-Hop Topology

In this case, a single-hop topology is investigated, employing 4 IPv6-enabled sensor motes. Figure 7.2 shows the results obtained from the experiment using 6LoWPAN REST, as well as the results obtained when considering simulated REST, without the IPv6 overhead. In both cases, the increase in response time as λ increases is almost linear, however with lesser increase for 6LoWPAN REST. In particular, for the case of 6LoWPAN REST, there is an increase of 30% for a 10-fold increase of λ from 0.2 to 2.0, whereas for simuated REST there is an increase of about 265% for the same 10-fold increase in λ. This observation is an indication that 6LoWPAN REST scales well as the traffic increases.

Still, simulated REST is of course much more efficient, especially in light workload. In this case, simulated REST is almost four times faster than 6LoWPAN REST. In heavy workload (e.g. $\lambda = 2$) the difference in performance is not so clear, being less than 1.5 times. This happens because the 6LoWPAN implementation based on Blip is more stable and reliable than simulated REST and operates faster. Hence, the performance difference is mainly due to the IPv6 burden during the message transmission/reception at the 6LoWPAN REST case.

Even though simulated REST outperforms 6LoWPAN REST in normal conditions, this is not the case when HTTP caching is used[1] (see Section 4.2.5). Examining Figure 7.2, we can also observe the results when the Web cache of the application framework is set for 10 seconds freshness time. In light workload scenarios, the Web cache does not influence response time

[1]Web caching is used for the 6LoWPAN REST case. A cache could also be employed for simulated REST, with effective potential improvements in response time. However, the main goal here is to evaluate the influence of Web techniques in IPv6-based smart home deployments, considering also the simulated REST option for better comparisons and understanding. Besides, a Web cache may be installed very easily by using a plethora of different available implementations on the Web e.g. Apache Cache Control, while it needs to be developed from scratch for a customized solution such as simulated REST

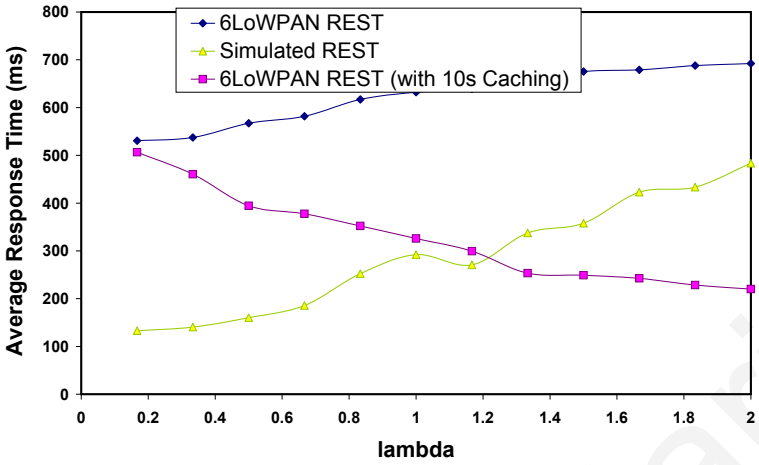

Figure 7.2: Response times in request-response interaction at a single-hop topology.

significantly, since cache hits are minimum. However, as the workload becomes heavier, the percentage of cache hits increases and the average response time is effectively reduced. In the special case when the arrival rate λ reaches 1.4, 6LoWPAN REST with 10 seconds caching needs less time than simulated REST. This time difference becomes 2.2 times bigger in the highest-traffic test where $\lambda = 2$. Furthermore, the decrease in response time in the caching-based case is almost linear as λ increases. In particular, there is a decrease of about 56% for a 10-fold increase of λ from 0.2 to 2.0. This is another indication that 6LoWPAN REST scales even better when Web caching is employed, even for minimal freshness times of some seconds.

The efficiency in response times using Web caching depends on the percentage of cache hits at each test. The correlation between cache hits and the arrival rate λ is depicted in Figure 7.3 (1 Hop). Higher traffic causes more cache hits since the application framework saves the responses in its cache, while answering requests from home tenants. Subsequent related requests, which occur

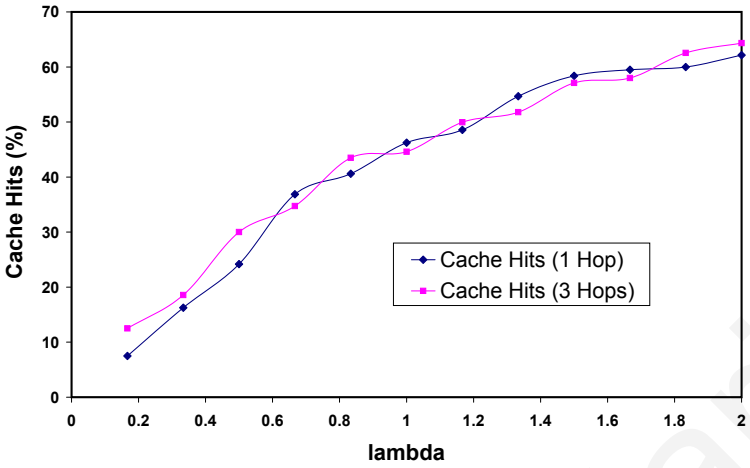

Figure 7.3: Cache success based on the arrival rate λ.

only seconds after a similar request has been satisfied (due to the increased traffic), are served through the caching mechanism.

Obviously, Web caching offers good scalability to the home ecosystem, both in terms of supporting numerous Web clients and sensor motes. Of course, cache hits and performance would have been increased if we used a larger cache freshness time. This depends on the specific requirements of each application. We will consider this correlation between cache hits and freshness time in the next scenario that involves a multi-hop topology.

7.2.2 A Multi-Hop Topology

In most cases, 6LoWPAN-based single-hop topologies do not cover adequately smart home environments, since the indoor range of a typical sensor mote is 20-30 meters. Perhaps, in the future, Wi-Fi technology would offer sufficient coverage using only a single access point [157].

Since singe-hop deployments are currently insufficient, in this section we evaluate the response times of our Web-based IPv6/6LoWPAN-enabled WSN also in a multi-hop setup.

The 4 IPv6-enabled sensor motes used in the experiments at the single-hop case, are placed now as leaf nodes in a 3-hop distance from the base station in a 2-3-4 topology, as shown in Figure 7.1. We note that this topology has been selected to stress test the operation of the WSN and the application framework in heavy workload.

The results of this experiment can be observed in Figure 7.4. The response times are less than a second, even in the highest-traffic test. The multi-hop environment causes additional delays of around 170-200 ms for 6LoWPAN REST and 110-140 ms for simulated REST, compared to the single-hop tests. In both cases, as λ increases, an almost linear increase of response times occurs, reaching 24% in 6LoWPAN REST and 127% in simulated REST, for a 10-fold increase of λ from 0.2 to 2.0, indicating a scalable behavior of 6LoWPAN REST.

Similar to the single-hop scenario, 6LoWPAN REST with 10 seconds caching executes faster than simulated REST, as soon as the arrival rate λ reaches and exceeds the value of 1. This happens in lower traffic conditions compared to the single-hop case, where λ needed to reach 1.4. A possible explanation is that the additional delays caused by the multi-hop environment make the caching mechanism more significant and effective.

Cache hits in correlation to the arrival rate λ, are shown in Figure 7.3 (3 Hops). Similar to the single-hop case, increased traffic produces a larger percentage of cache hits. As the cache hits increase, the response time is effectively decreased. This fact indicates good scalability of the system, also in the multi-hop topology, since numerous simultaneous Web requests can be handled in a few hundreds of milliseconds. Comparing with the single-hop case, we may conclude that cache hits do not depend on the sensor topology, being rather random for each different test, depending merely on workload.

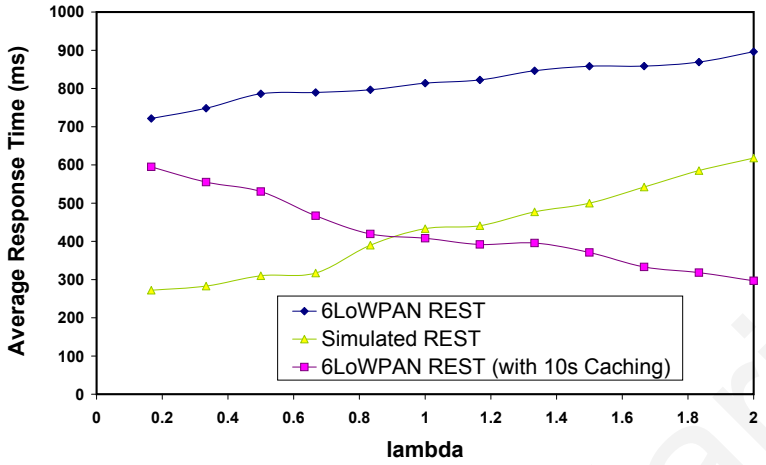

Figure 7.4: Response times in request-response interaction at a multi-hop topology.

Cache hits are directly related to cache freshness time. This statement is reinforced by observing Figure 7.5. According to the figure, as freshness time increases, the cache hits increase as well. This increase is higher between 10-25 seconds of freshness time and becomes smaller in larger values of freshness time (e.g. 40-60 seconds). In all the three scenarios involving different arrival rates, the graphs of the percentages of cache hits follow similar patterns. The percentage of cache hits may reach 90%, when considering freshness times around a minute. Obviously, this is a really high percentage that guarantees good performance in terms of response times.

A large positive correlation exists between the freshness interval and the successful request satisfaction through the caching mechanism. Definitely, there is a trade-off in using larger freshness time because although the cache hits are increased and the overall response times are reduced, the cached sensory measurements become older. The exact value of freshness time may be properly selected by considering the specific requirements of the smart home application scenario.

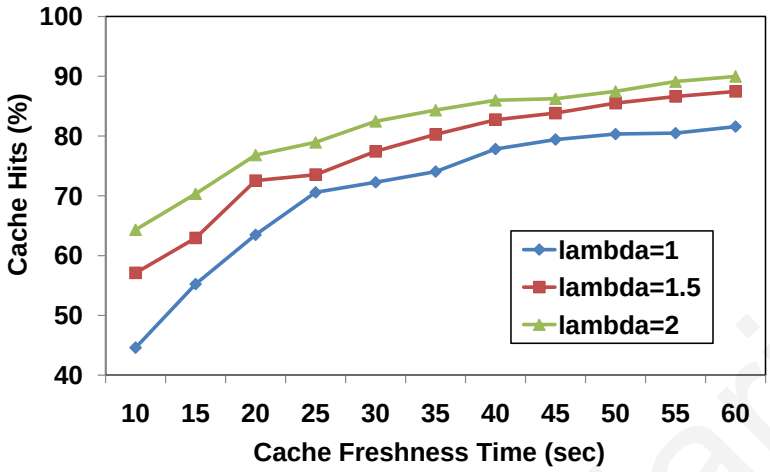

Figure 7.5: Cache hits in relation to the cache freshness time.

A more precise relationship between the cache freshness time and the response times may be perceived by viewing Figure 7.6. Apparently, as the freshness time increases, the average response time is effectively decreased. This happens since the percentage of cache hits increases, as we saw in Figure 7.5, and the framework serves the request through the caching mechanism instead of querying the physical home devices, which may be located multiple hops away from the base station. When increasing the default value of 10 seconds freshness time by 5-30 seconds, the request time is linearly reduced with a large gradient.

Further increases of the freshness time, beyond 40 seconds, cause only small further reductions to the response time. This observation is useful when considering the configuration parameters in specific smart home applications (see Appendix C), as the freshness time is much more effective when its value ranges between 10 and 40 seconds. Similar patterns hold for all the three tests using different arrival rates. As before, larger workload augments the significance and effectiveness of

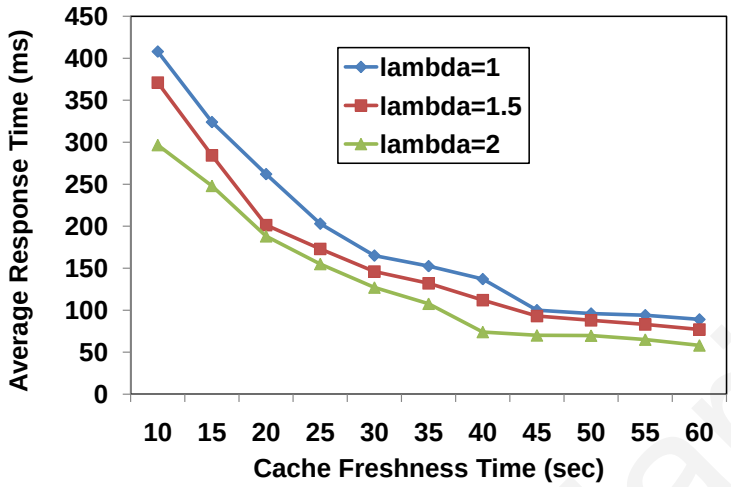

Figure 7.6: Response times in relation to the cache freshness time.

the caching mechanism. Overall, time-critical applications could exploit the caching mechanism to achieve minimum response times. As Figure 7.6 shows, freshness times of 60 seconds cause the average response time to diminish to only 100 ms, which is considered appropriate even for the most demanding smart home applications.

Finally, since the coverage of the wireless multi-hop network is important in large-home deployments, we decided to deploy our sensor motes also in larger distances from their parent nodes, in order to study the role of distance in response times. More specifically, we considered additional distances of 10 and 15 meters at the 2-3-4 topology of the 6LoWPAN REST WSN.

The results are shown in Figure 7.7. When the distance is increased by 5 meters, the response times are also increased by 65-109 ms. This accounts for an increase 8-12%. In case the distance is increased by 10 meters, response times are increased by 154-190 ms or 18-24% from the default case. Obviously, distance plays an important role in the performance of the system. However, even

in case of 15 meters distance and $\lambda = 2$, response times can be considered acceptable, reaching just a little more than a second. Definitely distance affects the communication properties of the network, by stronger effects of signal reflection, scattering and fading.

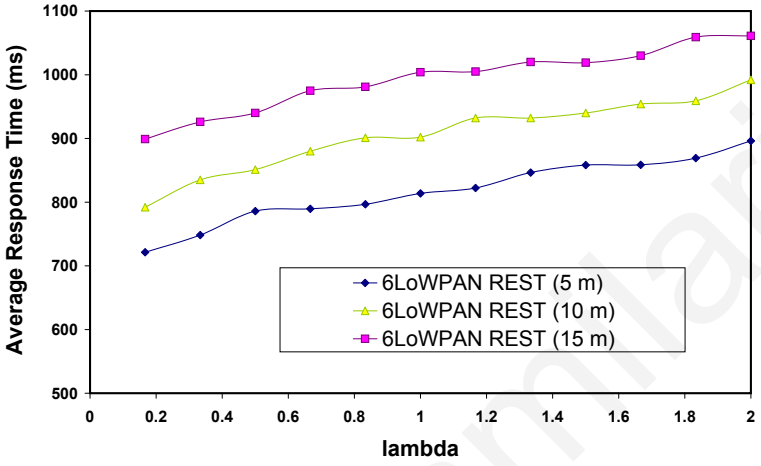

Figure 7.7: Response times in different distances at a multi-hop topology.

7.3 Energy Performance of Sensor Devices

In this section, the energy performance of 6LoWPAN REST in comparison to simulated REST is studied, by using the Avrora simulator [156]. Since Avrora simulates programs written for AVR microcontrollers, we adapted our code for Telosb sensors to run on Micaz motes. The power profiles of both mote types are listed in [134].

7.3.1 Energy Consumption in Request-Response Interaction

The energy-related evaluation efforts focused on the multi-hop tree topology of the sensor network, as it constitutes the most realistic scenario towards a real deployment in a smart home. Similar to Section 7.2.2, only the leaf nodes of the 3-hop topology were examined. Those are the sensors that satisfied the Web requests from home residents. We ran the same tests, assuming 5 meters node distance and 5 minutes execution time.

The energy consumption of a leaf sensor mote is displayed in Figure 7.8. The graph is for a single sensor but similar results have been obtained for the other leaf sensors. Furthermore, the simulation results for intermediate sensor motes in the multi-hop topology follow a similar pattern.

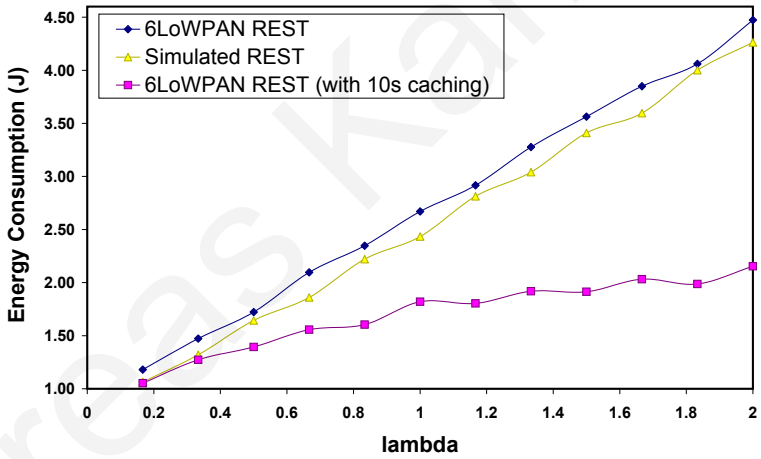

Figure 7.8: Energy consumption in request-response interaction at a multi-hop topology.

Observing the figure, we can observe that the difference in energy consumption between 6LoWPAN REST and simulated REST is small, around 2-12%. This variance happens due to the probabilistic selection of sensor motes by the residents. 6LoWPAN REST consumes more

energy due to the 6LoWPAN overhead in packet size (see Table 7.1). Both in 6LoWPAN REST and simulated REST, the consumption increases almost linearly.

When 6LoWPAN REST employs 10 seconds caching, it becomes much more energy-efficient than both 6LoWPAN REST and simulated REST, in workloads exceeding $\lambda = 0.5$. Energy efficiency reaches 50%, in heavy workloads. In light workload, efficiency is still considerable, fluctuating around 15-25%. This experiment indicates that 6LoWPAN REST with caching scales particularly well as the traffic increases.

We need to note that the results depend greatly on the transmission power setting. At some power settings, receiving (or idle listening) may consume more than transmitting. In this experiment, the link layer of the sensor nodes does not use any duty cycling, but it is always listening by default. The fact that sleep modes were not considered explains the rather increased values of energy consumption in the tests.

7.3.2 Energy Consumption in Event-Based Messaging

To test the proposed push technique for event-based Web messaging (see Section 5.2.4), a scenario was created, in which sensor motes informed the application framework as soon as the levels of illumination fell below a certain threshold. The framework was configured to POST a subscription HTTP request to the motes offering illumination services and supported the RESTful Message System [161]. Whenever the percentage of light in some room was significantly reduced, this event was POSTed back to the framework (subscriber) at a pre-specified callback URL.

We manually forced these events to be triggered, by turning the lights on/off. Considering the multi-hop scenario in Figure 7.1, around 400-440 ms were needed from the time the lights were turned off, until the event notification reached the application framework from a leaf node.

Obviously, in a pull scenario, event notification could last seconds or even minutes, depending on the polling frequency of the framework.

We used the Avrora simulator to compare the energy consumption of 6LoWPAN REST when RMS was employed (push technique, see Section 5.2.4) and when the framework periodically polled the sensor motes for newly sensed events (pull technique). We simulated the triggering of a fall in illumination levels once every 5 minutes. Therefore, the motes supporting RMS informed the framework every 5 minutes about these events while, in the pull case, the framework was forced to ask the sensors every 30 seconds about a possible new event.

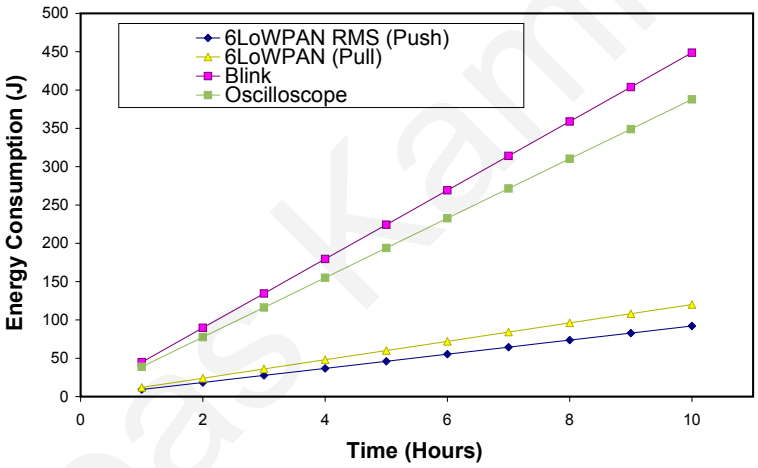

Figure 7.9: 6LoWPAN energy performance comparison in push- and pull-based messaging.

Figure 7.9 presents the results of this simulation, for some hours of operation. To provide a better overall picture, we also show the simulation results for Blink and Oscilloscope[2]. From the

[2]Blink and Oscilloscope are two well-known TinyOS applications. Blink is a simple application that blinks the 3 mote LEDs. It tests that the boot sequence and millisecond timers are working properly. The three LEDs blink at 1Hz, 2Hz, and 4Hz. Oscilloscope is a simple data-collection demo. It periodically samples the default sensor with 4Hz

figure, we can see that 6LoWPAN REST with RMS consumes considerably less energy than the classic pull technique. This difference is around 23%. Comparing with Blink and Oscilloscope, the energy consumption in the 6LoWPAN cases is 3-4 times less. This happens because blinking the LEDs of the sensor motes is very energy-demanding. More precisely, blinking the LEDs is responsible for 60-75% of the total energy consumed by the motes during their operation.

In general, our simulation measurements are linear. In real life this would not be true, as the battery would slowly discharge over time [100]. Thus, the supplied voltage would decrease, as well would decrease the consumed current and therefore the consumed energy. Unfortunately, the Avrora simulator does not take into account battery discharging. However, for 10 hours simulation time, the voltage drop is negligible.

Assuming that sensor motes operate with two typical AA batteries, they consume around 18,72 kJ. According to our simulation results based on Avrora, sensors with 6LoWPAN REST and RMS would be able to operate for 2 months and 23 days, while 6LoWPAN REST without RMS only for 2 months and 3 days. Under this assumption, sensors with Blink would operate for 17 days and 9 hours while sensors with Oscilloscope application for 20 days and 2 hours.

Nevertheless, through RMS push technique, smaller delays and better energy conservation are obtained, as we avoid polling the devices with continuous request messages. Of course, these results are dependant on the frequency of event reporting. However, some generalization can be made considering that events are supposed to happen only sporadically. We note that these calculations are not realistic since the link layer of the sensor nodes does not use any duty cycling for saving energy during sleep times. Though, the findings are useful for comparison purposes.

sampling rate and broadcasts a message over the radio every 10 readings, blinking at the same time the green LED of the mote.

7.4 Push Times in a Streaming-Based Interaction

In this last scenario, the IPv6/6LoWPAN-enabled WSN and the application framework were tested in a streaming application. The sensor motes were programmed to perform streaming of temperature measurements with frequency of 1 message per second. Since it was not adequate to test only the ability of the framework in terms of receiving those messages, we decided to extend the structure of the framework in order to provide enhanced eventing functionality.

We created a simple topic-based publish/subscribe infrastructure [56] with *push* [60] technology (see Section 2.6.2). RMS could also be used (see Section 5.2.4), however, RMS is more suitable for event-based communication between sensor devices.

Considering the experimental setup in Section 6.1.2, we used a second laptop, where we also installed the application framework, and attached to it a sensor mote acting as a base station. We settled this laptop near to the first one, connected to the same local-area network (LAN), in a distance where its base station would be able to sense the messages that were transmitted from the remote sensor motes to the first laptop. This second laptop, as soon as it started, it expressed its interest to the first, through a POST request, for events of topic *Temperature*. The extended experimental setup is shown in Figure 7.10. A single-hop star topology was considered, with each sensor mote in close distance from the base station, since the main aim was to examine the forwarding capability of the framework to interested third parties. Various sensor motes were used, ranging from 1 to 12.

The notion of *Web Hooks* (see Section 2.6.4) was followed to facilitate event notification. In the request payload, a *callback URI* was added, to indicate where possible events would be forwarded. From now on, whenever there was a transmission of a measurement about temperature, this message was considered as a new event and it was forwarded through the LAN from the first

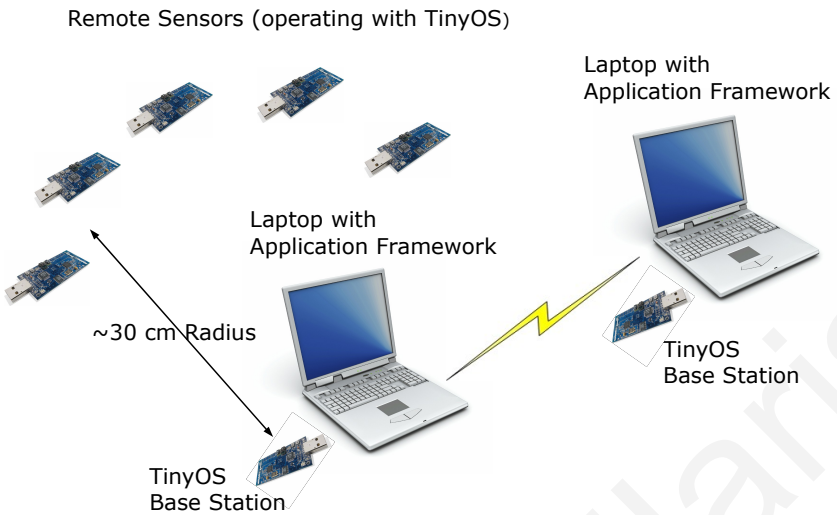

Figure 7.10: Experimental setup in the streaming scenario.

laptop (publisher) to the second (subscriber). The time difference between the moment the second laptop senses (through its base station) the event and the moment the very same event is received through the Web is measured and the results in regard to push time are presented in Figure 7.11, with a variable number of motes and correspondingly events/second.

From the figure, it may be inferred that the framework performs well also in the case of a streaming-based wireless network with reasonable workload. In all the cases, the delay increases almost linearly, which indicates limited scalability. In particular, all events need less than 60 ms to be pushed from the publisher to the subscriber. This time delay can be considered tolerable, even for emergency and highly time-critical deployments. Obviously, this delay would be larger if the subscriber was not in the same LAN as the publisher or if there was a multi-hop topology. Nonetheless, the results indicate the ability of the home ecosystem to handle a large amount of simultaneous streaming data effectively.

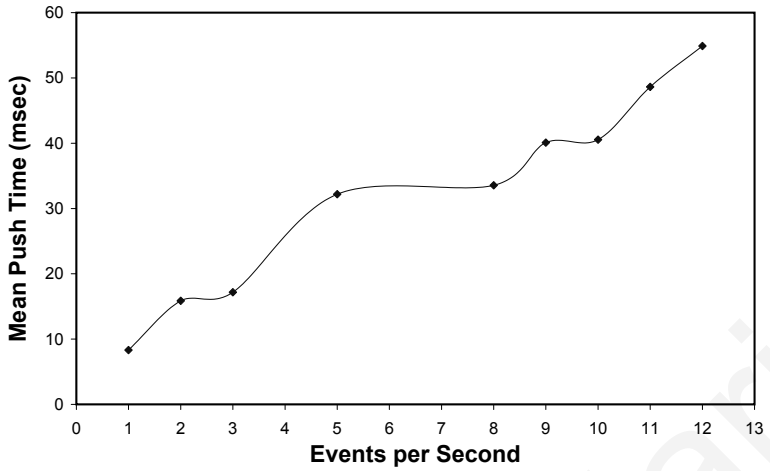

Figure 7.11: Push performance of streaming sensor measurements.

This experiment indicates that the application framework is scalable in terms of managing simultaneous streaming data from numerous sensor motes. Moreover, the framework has the capability of forwarding this data to an interested third party in minimum time and reliably (no message losses were observed). However, it would be also interesting to study the scalability of the application framework with respect to the number of subscribers, i.e. when numerous third-party applications may request this streaming data at the same time. For example, family members could observe the status of their smart home environment while at work from other Web applications, which would receive streaming data from the application framework.

To explore the support of multiple subscribers, we configured the second (remote) application framework to simulate Web clients that subscribe to the events produced by the (first) application framework. Hence, each streaming message would be forwarded to the second framework multiple times, but to different channels. These Web clients ranged from 10 to 80 and could be

parallelized with home residents. These numbers are considered satisfactory, even for large houses with numerous members. Three different tests were performed, involving 3, 6 and 9 sensor motes (and respectively messages per second).

Figure 7.12 presents the results of this experiment. In small numbers of subscribers (e.g. less than 50), push times grow linearly and the system scales well. However, as the number of subscribers increases significantly, the amount of time needed to forward the events to them grows with a larger gradient. In the specific case of 9 messages/second, push time grows almost exponentially. This means that the forwarding capability of the framework does not scale for hundreds of subscribers. Nevertheless, this is rarely the case in typical smart home applications.

Highly demanding scenarios could include 5-10 physical devices performing streaming of their sensory measurements and 10-50 interested tenants receiving those measurements. In such scenarios, the push-based forwarding performance of the application framework operates well, with push times less than a second. As the traffic increases, both in terms of sensor motes and subscribers, push times increase and may reach 3 seconds in the highest workload, which can still be considered a satisfactory amount of time in less critical applications.

We note that typical smart home scenarios would involve only few interested tenants, probably less than 10. In these typical cases, the performance in terms of event forwarding is considered very acceptable.

7.5 Summary of the Experimental Results

This section summarizes the results obtained during the evaluation procedure in this chapter. We note that the experimentation performed was not comprehensive, but nonetheless we will attempt to generalize our conclusions based on these limited experiments.

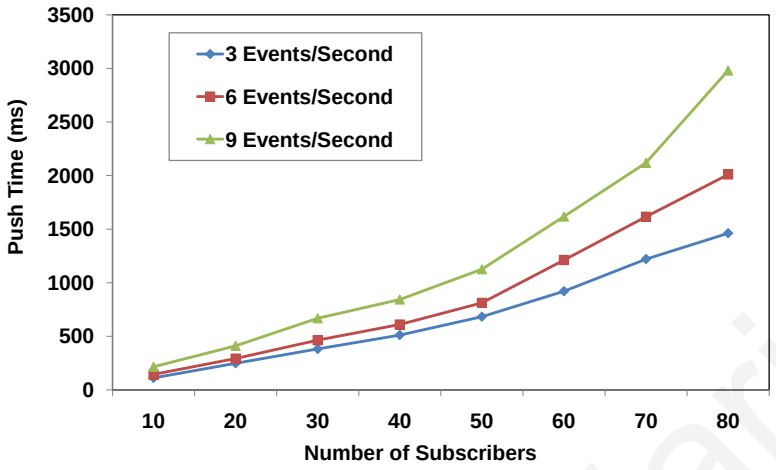

Figure 7.12: Push performance of forwarding streaming measurements to multiple subscribers.

A general finding from this effort is that the IoT is feasible for smart home deployments. Our experimental setup using IPv6-enabled sensor motes, operating with 6LoWPAN, behaved well and reliably, with acceptable performance in terms of response times and energy consumption. This performance of 6LoWPAN REST was comparable to that of simulated REST, which is a light, simulated version of REST without the IPv6 overhead.

Of course, simulated REST outperformed its opponent due to its optimized structure without the IPv6 burden. The situation changed when concepts of the WoT were employed, and especially Web caching. By using Web caching, the response times of the 6LoWPAN-enabled sensor devices were decreased significantly, falling below those of simulated REST when certain traffic was observed at the home environment. A similar finding was observed also when evaluating the energy consumption of the devices.

The experimental results are dependent on the specific sensor network topology. However, some generalization can be made, considering that the transition from a single-hop topology to a multi-hop topology of 3 layers of sensor devices did not affect significantly the results, which presented a similar pattern in both cases. According to the results observed, we believe that the application framework may support few tens of home devices without serious effects on the system's performance. Moreover, few tens of tenants may be reliably and efficiently supported, assuming that each tenant makes in average 1 request per minute to some of his home devices.

Another concept of the WoT which has the potential to improve the performance in some smart home applications is event-based Web messaging. Instead of polling continuously the sensor devices for possible events, events are pushed to the application framework only when something important happens.Through this approach, notification times were improved and the lifetime of the devices was extended.

Finally, we showed how the application framework behaved in a streaming-based interaction with home devices, when the streaming data needed to be forwarded to interested third parties, such as tenants using remote smart home Web applications. The experimental results showed that the forwarding capability of the framework was acceptable in typical smart home scenarios, but with scalability problems in large-scale applications involving hundreds of subscribers. Nevertheless, this is not the case for normal smart homes.

We need to note that the presented results are applicable for the considered scenarios. To increase confidence, further experimentation is needed, however, this comes at a very big expense in resources, human and otherwise. Future work will explore better the generalization of the results, as well as the scalability capabilities of the system, including different topologies and larger numbers of physical devices, mobility of devices inside the smart home environment, and support of hundreds or millions of simultaneous home residents.

Chapter 8

Blending Online Social Networking with Web-based Smart Homes

Online social networking sites have penetrated deeply in our lives, enabling collaboration and sharing on the Web. Two-thirds of the world's Internet population visit social networking or blogging sites, accounting for almost 10% of all Internet time [6]. Social networking has become a fundamental part of the global online experience, transforming the Web 2.0 into a social Web, in which human social capabilities are boosted. Currently, Facebook is the world's most popular SNS with more than 800 million active users.

The flexible Web-based architecture of the application framework enables the integration of online social networking sites towards the development of social applications that target home environments. The smart home functionality can be seamlessly blended with the operation of SNS, following the physical mashups example (see Section 4.2.6).

The chapter is organized as follows: Section 8.1 describes how sharing of home devices and services between online friends can be enabled through Facebook. After, Section 8.2 examines a social competition in blocks of flats towards energy conservation, exploiting social norms to influence people to save energy. Finally, Section 8.3 presents Social Electricity, which is a novel

Facebook application that allows people to understand their electrical consumption by means of social comparisons with their online friends, as well as local comparisons with their neighbors.

8.1 Sharing Home Devices through Social Networking

The popularity of online social networking motivated our efforts to examine the possibilities of leveraging existing online social networking infrastructures in order to transform the interaction with the smart home into a shared, social experience. Our goal is to extend the social relationships between people to social relationships with their home devices and services.

As the application framework exposes its functionality as a RESTful API, it is easy to develop a physical mashup that involves the household appliances of a smart home in a social networking application. The Web 2.0 technologies provided by SNS can be used to develop a smart home that presents social behavior. The general architecture for sharing home devices through online SNS is described in Section 5.2.5.

8.1.1 Implementation

We selected Facebook[1] as our experimental SNS because it has a stable open API that provides rich possibilities to application developers. We installed an Apache[2] HTTP server on a typical laptop, hosting the Facebook application. The application was implemented in HTML and PHP using the Facebook SDK for PHP library[3], in order to utilize Facebook open API[4] through PHP. The Facebook application's HTML code was enriched with Facebook Markup Language (FBML)[5].

[1] http://www.facebook.com/

[2] http://httpd.apache.org/

[3] http://developers.facebook.com/docs/reference/php/

[4] Open API is deprecated by Facebook, replaced by Graph API (http://developers.facebook.com/docs/reference/api/). Applications written with Open API are not supported anymore.

[5] http://developers.facebook.com/docs/reference/fbml/

FBML is an evolved subset of HTML with some elements added that are specific to Facebook and facilitate the building of social applications. Graph generation of the energy consumption of electrical appliances was achieved via Phplot[6], which is a simple graph library, written in PHP. Finally, a MySQL[7] database was installed on the Web server, serving as a permanent storage unit.

The Facebook application is split into six different interfaces. *Status* presents the latest measurements offered by available sensor devices deployed at the smart home (e.g. Telosb sensor motes), *Devices* lists all the connected devices, *Services* records the RESTful Web services offered by these devices, *Interaction* provides the means to users to interact with actuators (e.g. Plogg smart power outlets), *Eventing* allows the user to subscribe to events and finally *Profile* is the section where a user can manage his current event subscriptions.

In Figure 8.1, a snapshot of *Social Home* Facebook application is provided, in which the environmental conditions and the instant energy consumption of the electrical appliances inside the smart home are graphically listed. We may observe that the application is embedded to the user's profile *by design*. Whenever the user logs in his Facebook account, with a single click from his main Web page (*Wall*), he can connect to his home environment.

The owner of the application can share his home devices with his friends or his family easily, just by adding them in a *Facebook group*. Groups let people to share things only with their favorite contacts. For demonstration purposes, we created the Facebook group *Social Family*, which operates privately. Solely the administrator of the group (owner of the smart home), can approve requests for new members (residents of the house) to join the group. Only authenticated users in Facebook, who are authorized by the home owner to be members of that group, can fully

[6]http://phplot.sourceforge.net/

[7]http://www.mysql.com/

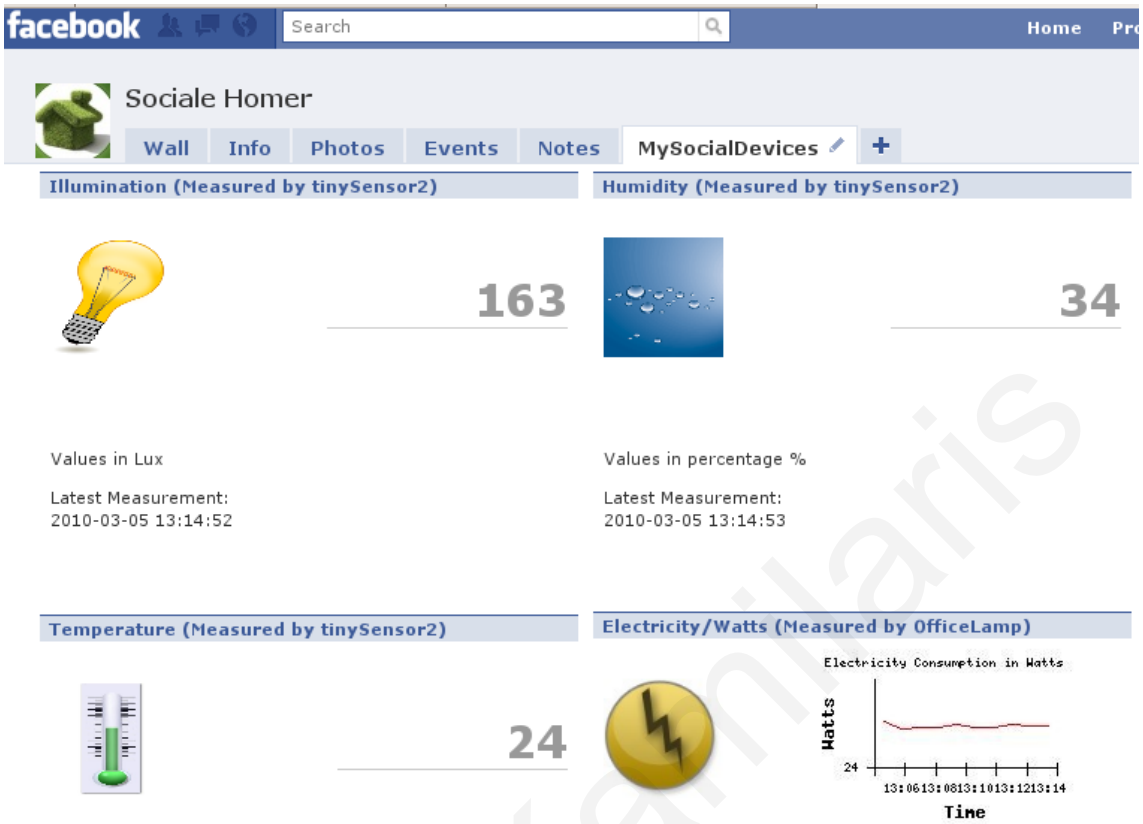

Figure 8.1: A snapshot of *Social Home* Facebook application.

manipulate the application. Therefore, user authentication and authorization is acquired with little effort, by reusing basic Facebook mechanisms.

Moreover, residents can subscribe to events, triggered when the environmental conditions inside the smart home exceed some threshold. For example, they can subscribe for events of type *Temperature*, when it exceeds 30 degrees Celsius for a specific timeframe. In such case, they can be notified by means of the notification mechanisms provided by Facebook API: through an update of their status; through a new post on their Wall; through a new note in their *Notes*; or through an email, sent to the email address with which they are registered in Facebook.

In Figure 8.2, the publication of a new event is displayed, posted on the user's Wall for the previous subscription example. A new measurement of *Temperature* of 33 degrees Celsius caused the validation of the user's subscription and the triggering of the appropriate notification mechanism.

Figure 8.2: A snapshot of the user's Wall with a notification about an event.

8.1.2 Benefits from Social Smart Home Applications

Numerous technical benefits may be obtained when combining the functionality of a smart home with online social networking. At first, the authentication mechanisms of SNS can be leveraged to achieve user authentication with just a few lines of code. User authentication would then allow the creation of authorization schemes for each user category. Some SNS even offer the possibility of creating private groups of users. Solely the administrators of these groups can approve requests for new members to join. Therefore, a role-based authorization mechanism can be built without much effort. For example, we exploited this functionality to allow the owner of the smart home to restrict access to his family members, in order to fully manipulate their house through the SNS application.

In addition, notification mechanisms can be easily developed, when using a SNS that supports notification triggering through its API. We demonstrated this possibility through a content-based publish/subscribe eventing infrastructure. With little effort, family members were able to subscribe to selective events by specifying some constraints. Users can be notified through various notification methods, depending on the available mechanisms of the SNS. For urgent notifications, users can even be informed through SMS, in case they register their mobile phones.

Through the proposed social approach, smart homes can be transformed into interoperable, shared spaces. Social Home Facebook application constitutes a challenge for various Web applications, to gain social perspective through integration with the social structures of SNS. For example, in a medical application scenario, such an integration would allow real-time, direct interaction between the doctor and his patients and it would facilitate the monitoring of critical parameters under a social environment.

As well, the practice of creating smart home applications with a social context, promotes sharing of physical devices and services between family members. Such applications could increase the awareness of residents about their home environment, helping them to acquire more sustainable behaviors. SNS users could find it easier to accept and use such applications, as they become blended with their overall online experience.

We believe that social networking infrastructures are mature enough in order to be utilized in general for giving a social shape to pervasive Web applications. We envision physical things as social entities just like human beings and we believe that their socialization will help them blend smoothly with the future Web. Perhaps the future would reveal a *Devicebook*, in which relationships between people would be extended to relationships with physical objects and services.

Evrythng[8] is a company already working towards this vision, aiming to design a Facebook for everyday devices and physical things.

8.2 A Social Competition in Blocks of Flats for Energy Conservation

Buildings consume a large proportion (around 70%) of the world's total electrical energy [57]. More than 30% of all greenhouse gas emissions can be attributed to houses and buildings. Hence, energy-saving initiatives need to examine the rational energy management in houses and buildings.

In general, people are willing and capable to adapt their behavior to energy-saving lifestyles if given the necessary feedback, support and incentives. Especially the influence of the community by means of comparisons with other people's consumptions, has the potential to drive residents towards a more persistent behavioral change [31]. Social norms can motivate people to question their behavior, if they discover it is not "normal" [12].

Moreover, residents generally tend to learn from their neighbors, and receive encouragement and support. Receiving daily feedback and taking energy-saving actions in a social context can increase people's effectiveness [51].

8.2.1 Implementation

To motivate people become more aware about energy and reduce their electrical consumption, we created a social competition between neighboring flats in large residential blocks.

Ploggs smart power outlets were employed (see Section 5.1.2), equipped with external current transformers for loads up to 100 Ampere, for acquiring the electrical consumption of each flat in real-time. A Plogg was attached to the mains meter of every flat, communicating wirelessly in frequent intervals (every minute) the electricity data in JSON format to a laptop computer that

[8] http://evrythng.net

received them by means of a Telegesis USB stick. We installed on the computer the application framework for smart homes (see Section 4.2). The framework was responsible to parse the electricity-related data, extract the important information and forward them to a Web server and a Microsoft SQL Server database. The Web server was developed in C# and ASP.NET, tightly coupled to the online database. The Web site included a forum, where people could communicate and exchange tips about energy saving practices and techniques. A Facebook application was also created to show real-time information about the competition, as well as a Facebook group, in which residents were encouraged to discuss about the study. The system infrastructure is shown in Figure 8.3.

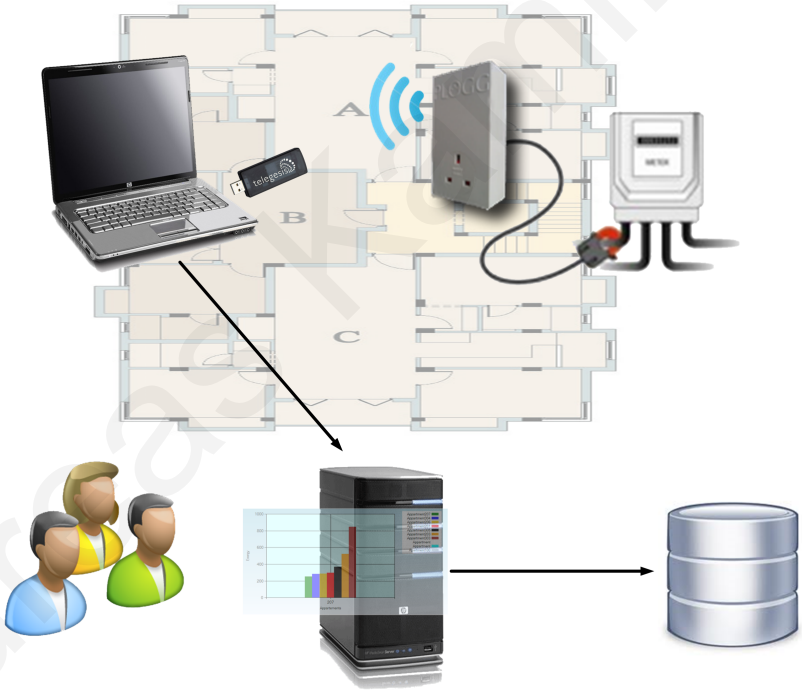

Figure 8.3: System infrastructure at the social competition in a block of flats.

Through the Web server, residents could authenticate themselves and get informed in real-time about their overall ranking in the competition, according to their electricity footprint. They could

also view their historical electrical consumption at the previous days of the competition, as well the overall electricity consumed by the block. All information about electricity is translated to money costs, based on the current tariffs of the electric utility.

Our aim was to motivate tenants to reduce their own energy consumption, but also to acquire energy awareness for the whole block, contributing to the overall reduction of energy. People generally respect the sense of social engagement and coherence and this helps them keep commitment over time [129]. In our case, the block's community becomes engaged to the goal of reducing the total electricity consumed. We did not want to inform people in real-time about their consumption, to focus our study mostly on the social competition as a way to save energy.

For those residents who were not familiar with the Web, we updated the ranking of the competition every day in a notice box located at the main lobby, along with brief reports about the energy performance of the building. We argue that when activities become pervasive, integrated into daily life, strong participation in social actions is more likely [108].

We decided to elect the winning flat, as the flat reducing most effectively its electrical consumption. To mitigate the possibility that residents could consume much energy in the first few days just to reduce it then and win the competition, we asked them for their electricity bills in previous months. By analyzing the bills, we had a complete report of each flat's energy performance.

Our award to the winning flat was a real-time energy monitor from Current Cost[9]. These smart meters are convenient in installation and use. The duration of the competition was one month for each block. After the competition, we distributed questionnaires to the residents to assess their commitment in the competition.

[9] http://www.currentcost.com/

Table 8.1: Age distribution of residents.

No.	Block	Flats	Residents	18-25	26-35	36-45	46-55	56+
1	Suburban	6	10	2	6	2	-	-
2	Urban	20	29	10	12	4	3	-

8.2.2 Case Study in Blocks of Flats

Our case study included two blocks of flats. The first is at a suburb, having 10 flats and the second in an urban area, having 20 flats. In the following subsections, the findings of the case study in these blocks are discussed.

8.2.2.1 A Block of Flats at a Suburban Area

The first block had 10 flats and was located at a suburban area, 10 Km from the city. Six flats accepted to participate in the competition, counting in total 10 residents. The ages of the participants are listed in the first row of Table 8.1. The total energy performance of the block, as a summation of the consumption of all the participating flats, is depicted in Figure 8.4.

Observing the figure, we can see the strong correlation of daily temperature to the energy consumption of the building. This is clear evidence that a considerable percentage of consumed electricity is utilized for heating.

Because of the high dependencies to temperature, safe conclusions about the energy savings due to the competition can not be extracted. However, we can just mention, by comparing the first two weeks of the study with the last two, that the energy consumption in the last two weeks is reduced by 260 kWh or 26%. Furthermore, in days with similar temperature (e.g. days 10 and 25), the energy consumption towards the end of the month is reduced by 22%.

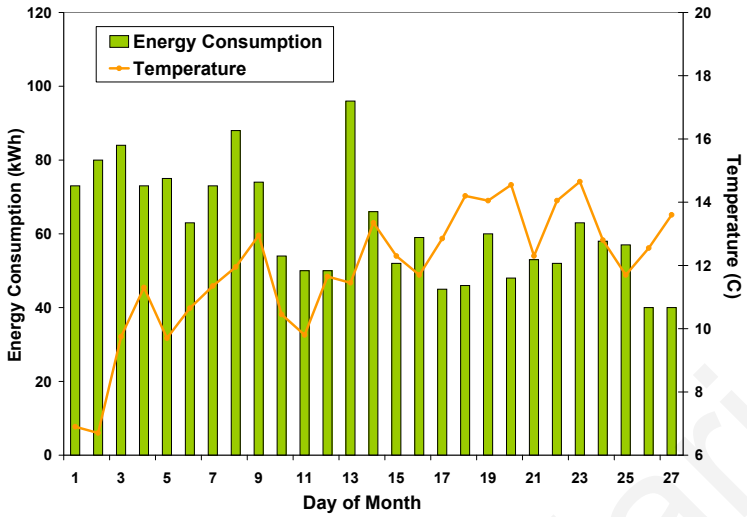

Figure 8.4: Everyday electrical consumption of the whole suburban block.

The study at this block was conducted in February, 2010 with average monthly temperature 11.60° Celsius and humidity around 63%. Comparing with the electricity bills of the flats in January, 2010, as shown in Figure 8.5, most of the flats have reduced their electricity footprint effectively. Flats located on higher floors needed more heating and consumed more electricity.

Considering that the average monthly temperature in January was very similar to February, around 11.56° Celsius with humidity 67%, we believe that the social competition has influenced most of the flats to reduce their consumption. The average reduction of energy is 11.90%.

The winners of the competition were a couple around 30 years old, living in flat 303, who reduced their electrical consumption by 41%! They both found this competition as a first-class opportunity to save money.

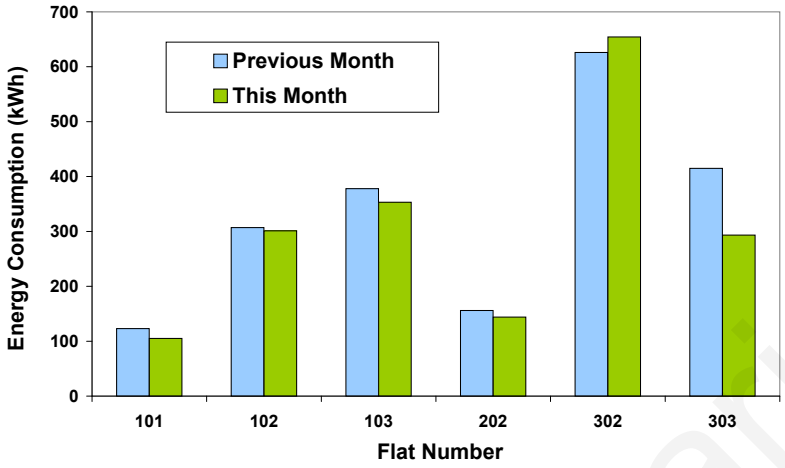

Figure 8.5: Consumption comparison of each flat with the previous month at the suburban block.

8.2.2.2 A Block of Flats at an Urban Area

The second block had 20 flats and was located in an urban area, near the city center. All flats, counting 29 residents, accepted to participate in the study. Most residents were young and educated people, and they were excited with the idea. The ages of the residents are listed in the second row of Table 8.1. The energy performance of the block is presented in Figure 8.6.

Also in this case, energy consumption is strongly correlated to temperature conditions. Comparing the first two weeks of the study with the last two, the energy consumption in the last two weeks is reduced by 1,091 kWh or 33%. Since in the second week of the competition there was a significant drop in temperature for 5 days, our findings can not be accurate. We can notice that in days with similar temperature (e.g. days 7 and 26), the energy consumption towards the end of the month is considerably reduced, of the order of 13%.

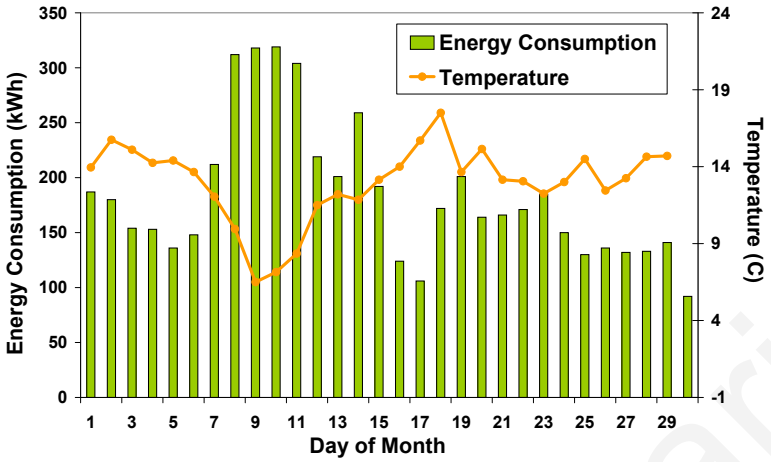

Figure 8.6: Everyday electrical consumption of the whole urban block.

This study was conducted in March, 2010 with average monthly temperature 13.00° Celsius and humidity around 59%. In Figure 8.7, the consumption of each flat is compared with the consumption at the previous month (February, 2010), as indicated at their electricity bills.

From the figure, we can observe that most flats have reduced their consumption significantly. Impressively, the average energy reduction reaches 27.74%. An important factor for this reduction is the average increase of temperature in this month by 2° Celsius. Once more, residents leaving on the third floor needed more heating and consumed more energy. A 31-years-old woman, staying at flat 303, was the winner of this competition. She managed to reduce her electricity footprint by 43%, and she was really proud of this, because she believed she helped protecting the environment.

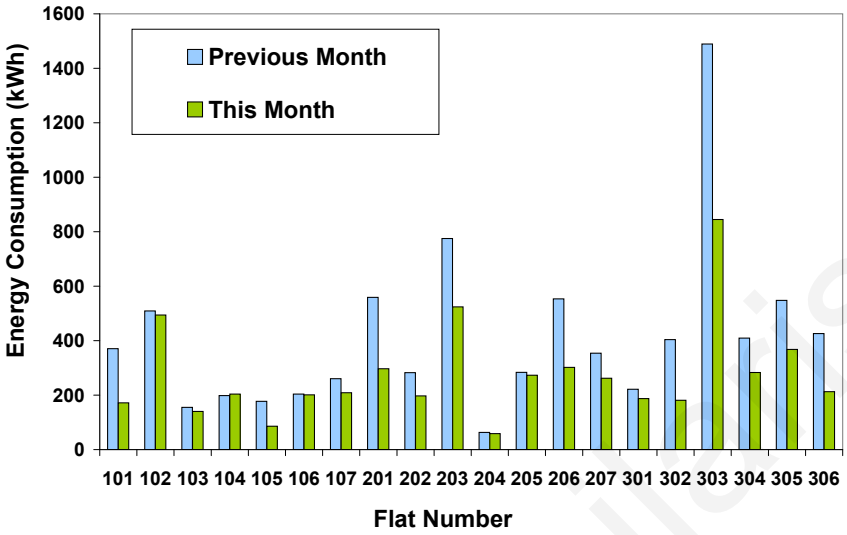

Figure 8.7: Consumption comparison of each flat with the previous month at the urban block.

8.2.3 Analysis, Statistics and Discussion

Our findings suggest that social competitions can influence residents to reduce their consumption remarkably. We discuss in the following subsections our general observations and experiences from the case study.

8.2.3.1 Suburban Vs Urban Block of Flats

Comparing the block at the urban location with that at the suburb, average energy savings in the urban case were 2.4 times more. People at the suburban block consumed in average 11% more energy compared to the urban block. However, temperature strongly affected this fact. Residents at the urban block were more positive about the competition. Since a large proportion of them

were highly educated students, it was easier for them to understand and accept the motivation and terms of the competition, inspiring also the other residents to pay more attention to it.

8.2.3.2 Demographic Analysis

In Figure 8.8 (left), we can study the residents' distribution of ages, in relation to their energy consumption. Examining the figure, older people (ages 46-55) consume more electricity[10]. This probably happens because they spend more of their time at home. However, this age group, together with the age group 26-35, are mostly influenced by the competition, reducing their consumption by 32%. It is more convenient for people that spend much time at home, to observe and analyze their consumption, taking countermeasures. For example, students may be sometimes busy with their studies, not having in these periods much free time in order to give a high priority for reducing their consumption.

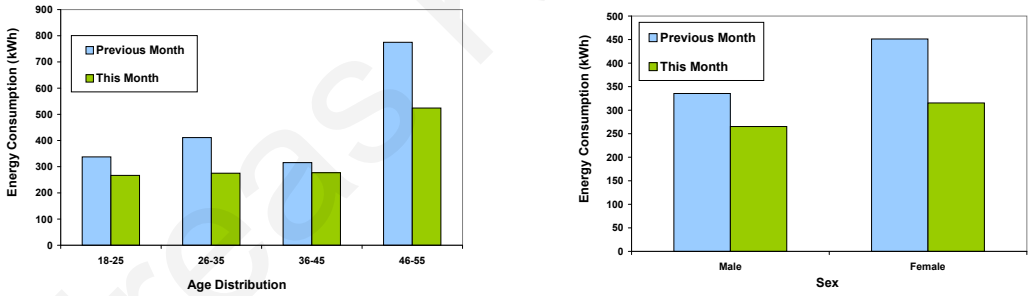

Figure 8.8: Electrical consumption Vs Age distribution of residents (left). Electrical consumption Vs Sex of residents (right).

Comparing the sex of the residents, as shown in Figure 8.8 (right), females tend to consume more electricity. This is logical, as they usually spend more time at home, having energy-demanding habits (e.g. washing and drying their hair, preparing food). Nonetheless, women

[10]We discuss this finding with some reservation because of the small sample of older people involved in the study.

contribute more in saving energy, reaching 30% reductions, while men around 20%. In general, females were more interested in the competition than males.

We then compared the electrical consumptions, according to the number of residents at each flat. This comparison is illustrated in Figure 8.9 (left). Obviously, more tenants at each flat implies more consumption. While this difference is more significant when comparing flats having one or two residents, reaching 44%, it becomes very small between flats of two and three residents, around 4%. Concerning energy-savings, flats with one tenant achieved most savings, nearing 30%. We believe it is easier for someone living alone to develop his own energy-efficient practices. These observations indicate that future energy-saving campaigns could place a high priority on one-bedroom flats, as the margin of potential savings is much bigger.

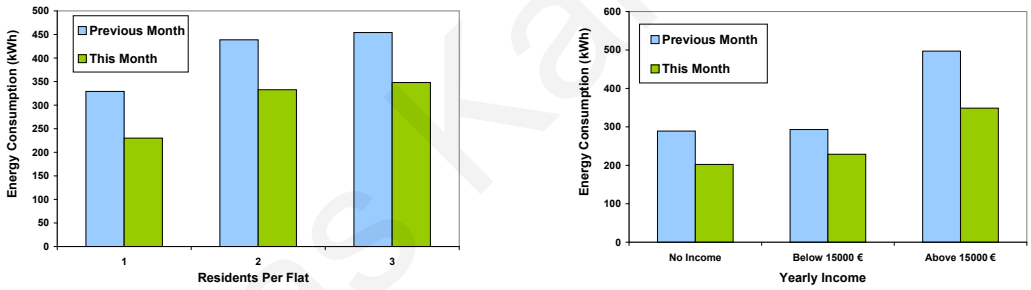

Figure 8.9: Electrical consumption Vs Number of residents per flat (left). Electrical consumption Vs Income of residents. (right).

Our final comparison, presented in Figure 8.9 (right), is about the income of the residents, in relation to their consumption. People with no income are students, financially-dependent on their families. As expected, tenants with high income consumed more energy. We assume that they are not willing to sacrifice their comfort just for saving money. On the contrary, residents with low income consume less than half the energy of their high-income neighbors. This group of people had the least savings in the competition, perhaps because they had already tried to

save energy in the past to reduce their costs. Even though residents with high income consumed increased electricity, maintaining high-comfort levels, their percentage of energy savings reached 30%. Most of them were motivated because of environmental reasons and not because of saving money. This observation suggests that future practices should try to influence people from the upper classes, by giving them incentives about protecting the environment.

8.2.3.3 General Statistics

Generally, 72% of the tenants stated that they were actively involved with the social competition, which helped them acquire a more sustainable lifestyle. 94% believed that this competition will influence them to save energy in the future. 69% of them consider that the method of comparing consumption with neighbors is a promising way for saving energy.

48% of people used the Web site for being updated about the competition. All residents also checked the information placed everyday in their notice boxes. Younger people preferred to use the Web in order to get informed, however, older people found much more practical the notice box. Perhaps, this inconvenience of older people using technology needs to be considered when researchers design energy feedback systems in the future.

Unfortunately, the online forum was not utilized by the residents for exchanging energy-saving tips. The Facebook application was used by six residents or 15% of people, who preferred to be informed about the competition while amusing through online social networking.

Asking the residents whether they wanted to be informed in real-time about their electricity footprint, 89% were positive. From them, 88% were willing to buy a product that would show them their energy consumption in real-time. They would invest at most 70 Euro for such a product. Some of them were surprised when we explained to them that this is possible at these costs. Some people did not even know that such products exist.

8.2.3.4 Suggestions from Residents

We also asked the tenants for suggestions concerning more effective feedback about energy consumption. The most popular suggestion concerned feedback through SMS, sent by the utility once per day. Daily feedback through email was another interesting suggestion. Some people requested that the electric utilities should provide more detailed electricity bills. Still, 61% believed that a social competition is more effective than what they suggested.

Finally, all residents agreed that the government must give smart incentives to people to save energy. According to them, these incentives could be similar competitions with awards from the utilities, scalar pricing schemes that reward green flats and houses while punishing energy-wasting buildings, more pervasive and real-time energy feedback techniques and grants from the utilities or the government for renewable energy systems and green lighting.

8.2.3.5 Discussion

Our findings were very positive, however, they just constitute the product of a small case study and it can not be considered safe to extract conclusions about the general population. Furthermore, the selected months of the study have prevented a solid analysis of the results, due to large fluctuations of temperature. Future studies need to select months in which temperature is not as dominant factor in the energy consumption of residents.

A main drawback of our approach is related to the necessity of having costly real-time energy monitoring devices installed in the flats, which is not a common practice nowadays. Nevertheless, the smart grid[11] suggests the massive deployment of smart meters in residences (see Section 2.1.2), for real-time acquisition of household electrical consumption. Hence, the practice of social competitions complies with the future smart grid, as the Ploggs smart power outlets we employed

[11] http://smartgrid.ieee.org/ieee-smart-grid

could be replaced by residential smart meters. Accordingly, such social competitions for saving energy could be enabled by the electric utilities in the near future.

This study conforms to the broader vision of smart neighborhoods, defined as groups of buildings (e.g. blocks of flats) mimicking living systems, collaborating inside the future smart cities. Similar to living systems in nature, buildings will act collectively in relation to the ecosystem (e.g. neighborhood) they reside in. This will help to address safety, health-related or sustainability issues in the city, such as energy conservation and reduction of carbon emissions.

8.3 Energy Awareness through Social Comparisons

Smart meters are currently being deployed, or planned to be deployed, worldwide. One of the expected promising benefits is to provide timely overall electrical consumption feedback, helping people to become more energy-aware and control their electricity footprint. However, still people can not clearly consider quantitatively how much energy they consume. They do not possess the proper metrics to define whether their total consumption is low, average or high. This happens because each area, city or country has different tariffs and varied weather and physical conditions.

A promising way to understand the semantics of consumed energy is to compare it with the amount consumed by relatives, friends and neighbors. In this case, SNS are the ideal platforms for such comparisons, as they maintain a highly accurate graph of the social networks of their users.

8.3.1 Social Electricity Facebook Application

To demonstrate whether energy awareness can be reinforced by social comparisons, we created *Social Electricity*[12], which is a Facebook application that allows people to perform comparisons of their electrical consumption. Social Electricity aims to make citizens aware of their

[12] https://apps.facebook.com/socialelectricity/

energy consumption, by means of comparisons with the corresponding electrical consumption of their friends, as well as with the total consumption in the street/neighborhood/village/city/country where they live. By effective and realistic comparisons, the consumers may perceive their energy behavior and take steps to reduce their electricity and carbon footprint.

We collaborated with the Electricity Authority of Cyprus (EAC)[13], which is the only electrical utility in Cyprus. EAC delegated us with anonymous access to the domestic consumption in Cyprus, categorized according to the address of their consumers. This data includes the measurements of all electricity meters deployed in domestic premises around the country, collected every two months. More frequent updates are yet not possible, because EAC has still not upgraded its electrical meters to smart meters (see Section 2.1.2).

Respecting the privacy of Cypriot citizens, the electricity measurements were aggregated at a street level (address, postal code, city). Thus, it is not possible to derive the analytic consumption of some specific residence. However, residents are encouraged to add by themselves their exact monthly electrical consumption, and compare it with their local neighborhood or their friends.

A snapshot of Social Electricity is provided in Figure 8.10. As we can see from the figure, users can see their friends and their associated energy consumption in their current place of stay, tagged on the map of Cyprus,. When people connect to the application for first time, they are asked to select their address, offering them the option to keep their place of stay private, if they wish. Friends with higher consumption are displayed in red color while friends with better energy behavior are displayed in green.

In general, Social Electricity supports the following functionalities:

- It allows people to compare their electricity footprint with that of their neighborhood/village/town, to perceive if their own consumption is low, average or high.

[13] http://www.eac.com.cy/

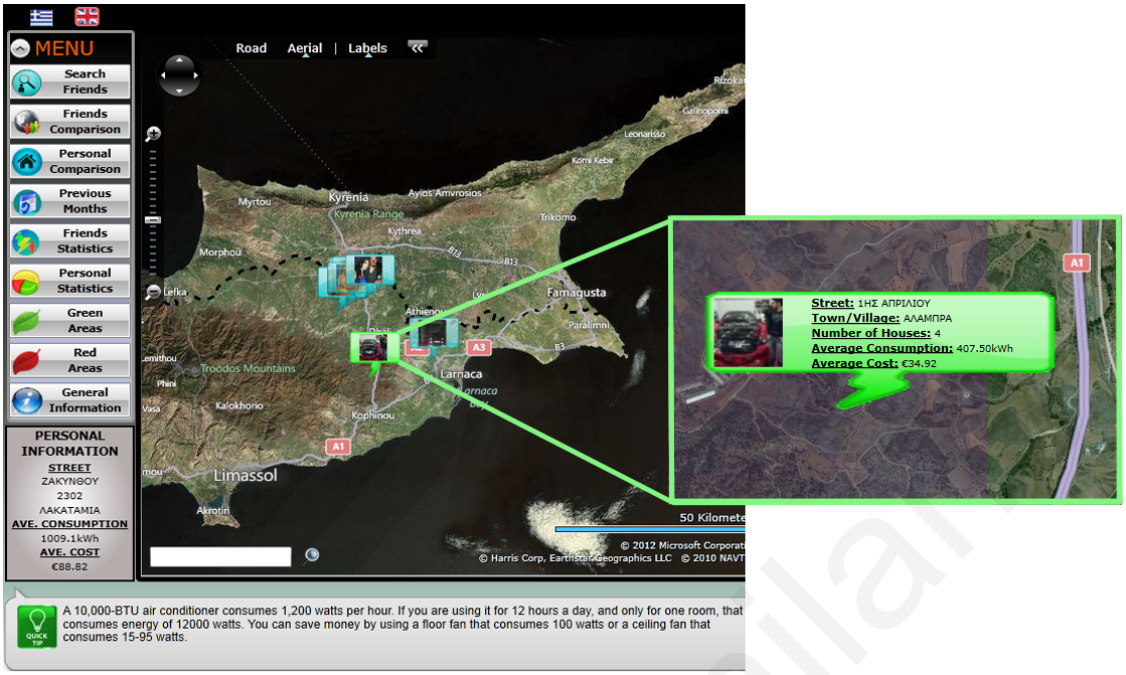

Figure 8.10: A snapshot of *Social Electricity* Facebook application.

- It associates electrical consumption with actual costs, enabling users to have a more meaningful view of their energy profile.

- It promotes sharing of peoples' electricity consumption figures with their friends at a street level. It can transform the procedure of saving energy into a social game.

- It gives useful tips to people in order to reduce their energy consumption and increase their overall environmental awareness.

- It provides useful statistics about the most energy-efficient streets, villages and cities near the user's location. Thus, it helps people to acquire region awareness, inspired to take actions for helping the local community maintain a better future ranking. This capability is depicted in Figure 8.11.

- It offers statistics about the least efficient areas near the user's place of stay. Hence, people may start questioning their behavior, in case they discover that their area is energy-hungry.

- Statistics are given concerning the areas with the least/most energy consumption in the whole of Cyprus. Users may select the province or minimum/maximum number of houses, and view statistics about the least/most energy-efficient villages of the country.

- Users may observe the electrical consumption of their street in previous months and compare it with the present street consumption. They can perform this historical comparison also in relation to their Facebook friends, to observe their friends' street energy behavior and compare it with the behavior of their street.

- Finally, users may examine their street's consumption in the same month in previous years and compare it with the current street consumption. This comparison may also include the street energy behavior of their friends. This option is shown in Figure 8.12.

Social Electricity aims to be extended into a healthy competition, where each user competes with his friends for less consumed energy. People from the same area are encouraged to cooperate to reduce the overall consumption and improve the green ranking of their area.

We need to note that without the existence of SNS, pervasive applications like Social Electricity, which require social content, would not be practical. It would be extremely difficult to acquire the social networks of people. In addition, promoting the application would be a time-consuming and expensive process.

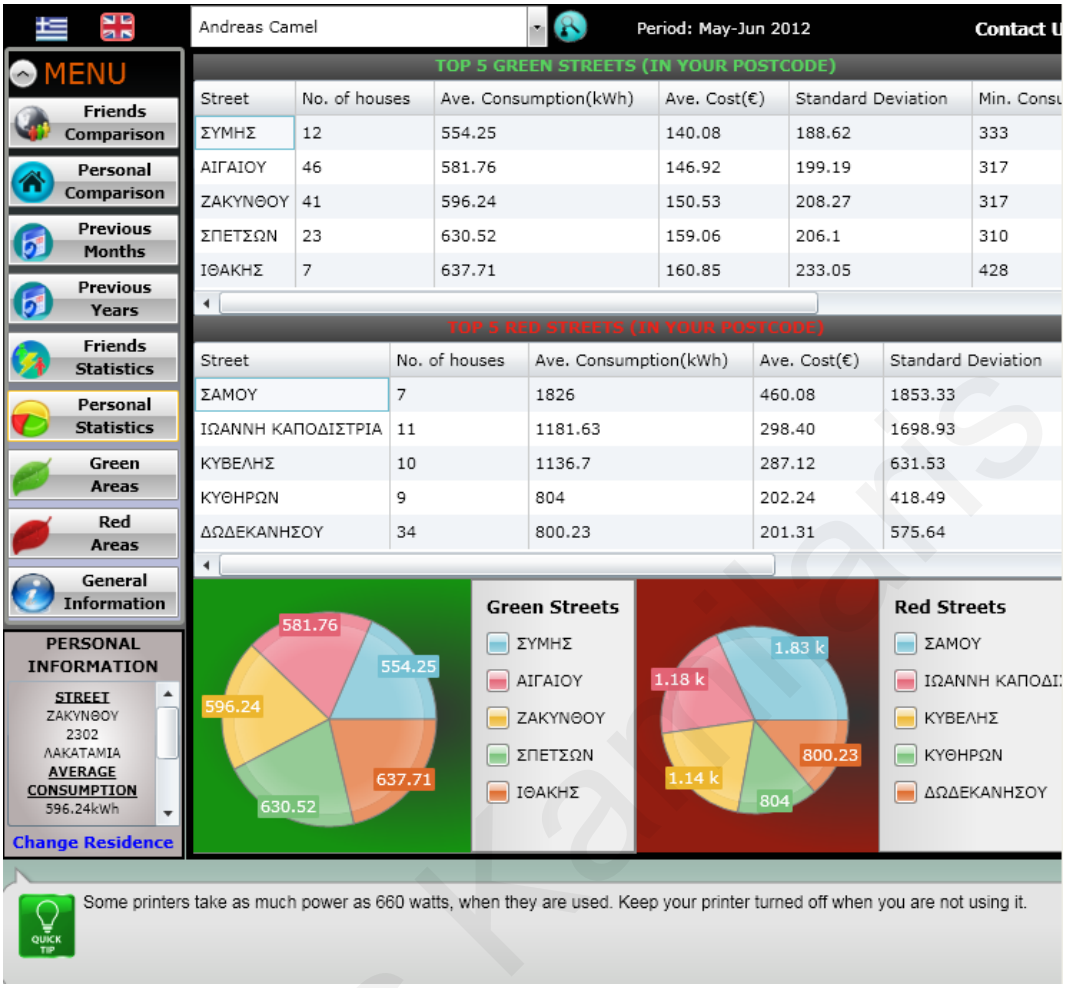

Figure 8.11: A snapshot of *Social Electricity* showing local electricity statistics.

8.3.1.1 Implementation

The Web server hosting the application was developed using Visual Studio 2010 and Microsoft Silverlight[14]. The electricity consumption data have been stored on a Microsoft SQL server while the connection between the Web server and the database server has been achieved using WS-* Web Services [13]. Bing Maps permit people to view on maps their friends, in the location where they live, together with the average energy consumption in their locations. Translation of street

[14] http://www.microsoft.com/silverlight/

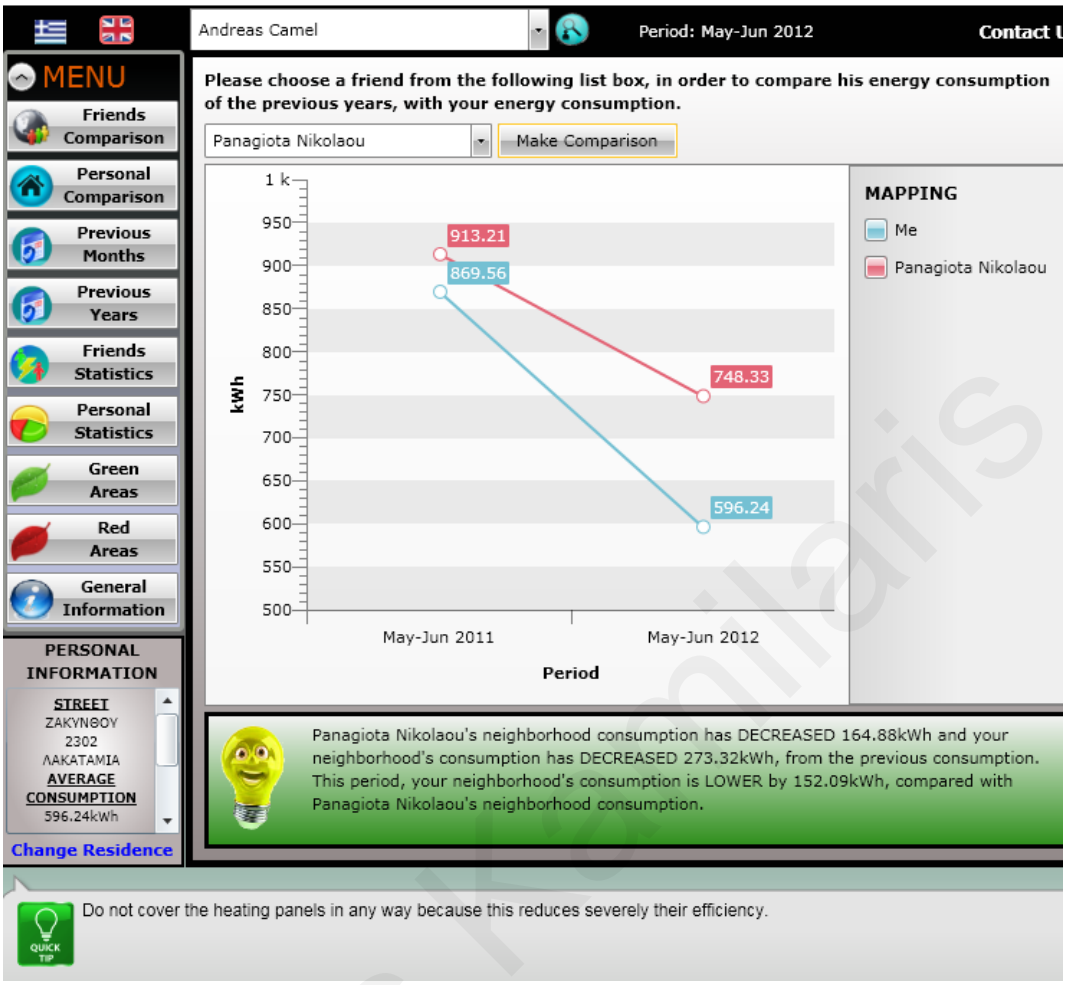

Figure 8.12: A snapshot of *Social Electricity* showing historical energy information of the same month in previous years, in regard to the user's street, comparing also with a friend's street.

addresses into absolute physical locations (longitude, latitude) has been enabled by the valuable help of GeoPostcodes[15].

8.3.2 Preliminary Evaluation

Primarily, Social Electricity was deployed locally, at the Computer Science department of the University of Cyprus. The initial sample comprised of 72 students of the department, who

[15] http://www.geopostcodes.com/

were asked to use the application. The majority of them already had a Facebook account, since Facebook is popular among students. Two weeks after the release of the application, we asked the participants about their impressions. The methodology was elementary, involving only small chats and personal interviews.

Not surprisingly, people were impressed by the capabilities of the application. They found it really entertaining to compare their local energy consumption with that of their friends. We even listened to some students teasing their colleagues about the poor energy performance in their street. We considered this teasing as a positive step towards saving energy in the future.

Social Electricity becomes useful only when people enrich their Facebook profile with information about their place of stay. Some students complained that their friends did not appear on the map as they did not fill in yet their (general) address details. However, this phenomenon occurred only at the first 2-3 days, before a large number of students utilized the application.

Comparing their own consumption with that of their neighborhood allowed students to quantify their electricity footprint more precisely. 70% of the students reported that Social Electricity helped them become more aware of the energy they consume. Around 18% admitted they had a high electricity bill and they promised to consume energy more rationally in the future. One student particularly, realized after using the application that he had a faulty electricity meter. He contacted EAC immediately to replace it.

The initial success of Social Electricity might have depended on the sample's young age, which covers students 17-25 years old. A large deployment (see Section 8.3.3) would include people from other ages, whose friends might not be actively involved with Facebook. Nonetheless, these initial findings were encouraging to proceed with a deployment around the whole of Cyprus.

8.3.3 Large-Scale Deployment

Social Electricity was made available to the public officially on the 1st of August, 2012. This timing was selected because of the sensitive summer period, when the electricity demands in Cyprus are at the highest. Thus, our aim is that the application will influence Cypriots to consume electrical energy more rationally. In collaboration with EAC, we started monitoring the energy behavior of Cypriot citizens to consider whether this approach has contributed in increasing their energy awareness, and how much it has influenced them to save energy and money.

In September 2012, the application was awarded the first prize in the 2nd ITU Green ICT Application Challenge[16]. This prize is considered particularly important, taking into account the status and size of this organization (193 member countries and over 700 members from companies and academic institutes).

This success facilitated the promotion activities of the project, since extensive details about the award and the Facebook application were published in local newspapers, Web blogs and magazines, radio and the TV. A TV channel even organized a full reportage[17] dedicated to the project, explaining to people how to use it. After only 4 months, the application has been used by more than 1,000 Cypriot citizens (in a country of 300,000 domestic premises) and its Facebook page[18] has more than 1,300 friends/supporters.

Current work on the project includes the development of mobile applications in iOS (iPhone mobile phones and iPad tablets) and Android-based mobile phones. These applications are expected to broaden our user community. After numerous requests from Cypriots, we decided to develop a version of Social Electricity without the dependency on Facebook, in order to accommodate Cypriot citizens who do not possess a Facebook account. This version of the application

[16] http://www.itu.int/ITU-T/climatechange/greenict/201206/index.html

[17] http://www.ant1iwo.com/oikonomia/2012/08/11/to-magiko-application-toy-reymatos-video/ (in Greek)

[18] https://www.facebook.com/SocialElectricity

will be simpler, containing information about local energy consumption as well as general statistics about the consumption around Cyprus.

We also started sending personalized newsletters to our users, whenever there is an update of the electricity consumption data, to give them stimuli to reuse our application. These newsletters contain information about their latest consumption, in comparison to their respective consumption before a year, accompanied with a happy/sad emoticon as depicted in Figure 8.13, in case their consumption has been decreased/increased accordingly. Concerning our latest newsletter (November, 2012), from the category of people whose consumption was increased, 30% of the users checked the application when they received the newsletter. while this happened for a percentage of 32% of the users whose consumption was reduced.

Figure 8.13: Happy/sad bulb sent to the users of *Social Electricity* in a monthly newsletter.

In collaboration with EAC, we started examining various ways to further promote the application among the Cypriot consumers of electricity. Our future actions concern the sending of an informative leaflet, accompanied with the electricity bill of each domestic premise, embedding the application on the official website of the authority and giving some awards to local communities and villages that effectively reduce their overall electrical consumption. These awards could involve lower billing of the local consumers, prizes and small gifts to people, such as real-time energy monitors. Future activities in regard to the project are listed in Section 11.5.

8.3.4 Evaluation

A full evaluation of the project is currently underway. This evaluation includes both questionnaires, sent to the citizens who are using the application, as well as focus groups of people who would be encouraged to discuss different aspects of Social Electricity. Focus groups would be divided in the following 4 categories:

- Students who live in Cyprus with their parents and do not pay any electricity bills.
- Students who live in Cyprus alone and pay their electricity bills by themselves.
- Low-income citizens who work and live in Cyprus.
- High-income citizens who work and live in Cyprus.

The foundational pillars of the evaluation include the following themes: usefulness, effectiveness, acceptance, motivations, privacy and potential. Based on these pillars, the evaluation procedure aims to assess the significance and potential of the project. In parallel, a statistical analysis of the energy conservation achieved will be performed, focusing in areas in which the users of the social application live.

The first part of the evaluation procedure, namely the distribution of questionnaires to current users of the application has already been accomplished, In total, 190 subjects participated in the study. To increase the overall participation, we offered 6 energy meters to some of our subjects after a lottery. We present here the feedback and reaction of the subjects, in regard to our questions.

8.3.4.1 Usefulness and Potential

From our 190 subjects, 44% found the application very useful, while 36% just useful. In total, 80% believe in the usefulness of Social Electricity and this is a very positive fact. 55% of the

users were positively affected by the application to become more energy-aware. Probably these are users who realized an increased electrical consumption at their home, in comparison to others.

A percentage around 16% believed that their energy consumption is high, in comparison to their neighborhood and their friends. 61% believed they had an average consumption while 23% a low consumption. The percentage of users with low consumption is larger than those with high, because in general people that engage in such applications are more environmentally-aware.

78% of the users declared they perceived their energy behavior through the application and understood whether they consume low, medium or high amounts of electrical energy. A significant percentage of 62% claimed that their consumption was reduced in regard to the same period last year. This is another positive fact indicating the usefulness of Social Electricity, however, we need to take into account that there is an overall decrease in electrical consumption around Cyprus, due to the financial crisis. This decrease is around 10%.

57% are aware of their ”green” and ”red” friends, as they appear on the map of Cyprus, tagged where they live. This is a sign that a fruitful discussion for energy conservation may have occurred between them. The incentives of using the application are the following (the subjects could select multiple options): 48% from curiosity, 71% for preserving the environment, 71% for financial reasons and only 14% for their responsibility as citizens.

We also asked the subjects for their opinion about the most effective incentives for energy reduction. The most popular answer was a discount on their electricity bills, offered by their electric utility. Other suggestions included more frequent feedback for their consumption and competitions for energy savings.

Concerning the relation between the usefulness of the application with time, 61% believe that the application is mostly useful today. 28% believe it will be more useful in 1-2 years, while 11%

in more than 2 years. It is worth-mentioning that 39% envision the usefulness of the application in the coming years, probably in combination to the introduction of smart meters in our lives.

Impressively, a percentage of 64% believe that Social Electricity will help them to reduce their electrical consumption by more than 10%. 10% believe in 1-5% reductions and 26% in 6-10% decrease of consumption. It is highly encouraging that our users are convinced that the application will contribute in their strategies for conserving electricity in the future.

Interestingly, 69% agree that this application could be extended for comparisons in inter-country level, between citizens of different countries. Related to this, 63% are interested to compare their electricity footprint with their Facebook friends who reside in another country. Such comparisons could reveal behavioral patterns and attitudes of people from different cultures.

8.3.4.2 Privacy

A large part of the questionnaire was devoted to privacy aspects, as they constitute crucial issues in Social Electricity because of the sharing of personal information among Facebook friends. 69% of our users are willing to share their personal consumption among their friends, towards more effective comparisons. 77% believe that the application respects their privacy. However, the rest 23% is an important group of people who have some concerns about the overall privacy.

More specifically, 30% are concerned with the possibility that information about their energy consumption in neighborhood level could be revealed to third parties. People are also concerned with some specific features of the application. For example, 22% believe their privacy is affected by showing the location where they live on the map of Cyprus to their online friends. 15% are concerned about sharing the consumption of their neighborhood with their friends and 11% about sharing historical data of their neighborhood's consumption.

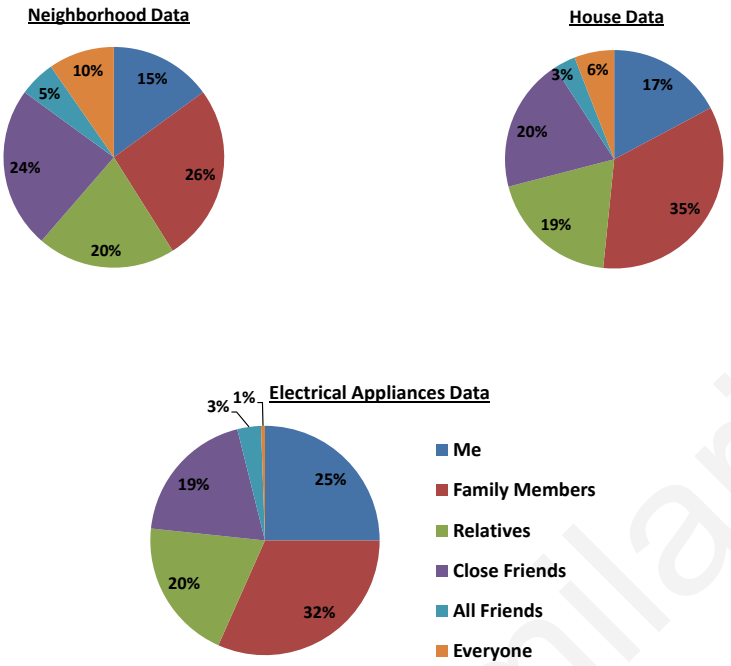

Figure 8.14: Privacy concerns of users of *Social Electricity* in regard to with whom to share their electricity data in neighborhood level (top-left), house level (top-right) and detailed consumption of their electrical appliances (bottom).

Obviously, a large group of users classify privacy issues as important in terms of sharing their electrical consumption among their friends. To identify the ”tolerance” levels of our users in regard to sharing electricity data, we asked them to inform us with whom they are willing to share the electrical consumption of their neighborhood, that of their house as well as the detailed consumption of their home appliances. The different available categories of people for sharing this data are: only me; family members; relatives; close friends; all friends; and everyone. Their answers are depicted in Figure 8.14.

As the results show, users have different sharing preferences depending on the sensitivity level of their electricity data. While around 20% are willing to share their electricity data with relatives

in all the three cases, they are reluctant to share this information with all of their friends or with everyone else. A large percentage of users trust the other members of their family and 32% are willing to share with them even details of their electrical appliances' consumption. A smaller percentage of 19% wish to share their detailed consumption with close friends and this percentage is increased in house or neighborhood data. This means that some users do not trust their close friends, in order to share with them their personal electricity footprint. It is remarkable that from the more general to the more specific consumption data, an increasing percentage of users trust only themselves for viewing these values. This percentage starts from 15% in neighborhood level and increases up to 25% in level of electrical appliances.

Summing up, the results from our online questionnaires are positive and encouraging about the usefulness and potential of Social Electricity. Privacy concerns constitute an important parameter we need to pay attention, since our users are reluctant to share their personal information with their online friends or with everyone. We plan to dedicate significant effort in improving privacy aspects in the coming enhancements of the application. We suggest that an effective solution for ensuring privacy would be to adopt an approach similar to that of Google+[19], according to which users could categorize their friends in "privacy circles". Then, users would be able to associate varying sensitivity levels with each of their circles, and share different aspects of electricity data with each of them (e.g. neighborhood data with all friends, home data with close friends and relatives, electrical appliances with family members).

[19] https://plus.google.com/

Chapter 9

Integrating Web-Enabled Smart Homes to the Smart Grid

Increasing energy demands, depletion of natural resources and rising costs make energy conservation a universal problem with tremendous environmental, political and social implications. Predictions denote that by the year 2030, the global energy demand will double, rising up the energy-related green gas emissions by 55% [89].

This high energy demand cannot be accommodated by current electricity grids. Most of the electricity grids around the world have been built many decades ago, to meet the energy requirements of the society at that time. They are being incrementally upgraded, but these upgrades may not be completely adequate for the future grid, in addition to the green concerns.

Increasing demand and environmental concerns influenced initiatives towards a more rational utilization of electrical energy. This goal can be best achieved when the electric utilities are fully aware in real-time about the electrical consumption and the demands of their customers. The grid becomes intelligent when it manages to deliver electricity from suppliers to consumers, supported by two-way digital communications and a smart metering system, in a fault-tolerant, secure and more reliable manner. This vision is believed to convert traditional electricity grids into modern

smart grids. A smart grid[1] is a network of networks that has come to describe the future electricity grid, enhanced with ICT and smart metering, applied to generation, delivery and consumption of electric power.

Electricity smart metering (see Section 2.1.2) involves measuring the electrical consumption of a house in frequent intervals and communicating that information at least daily back to the utility for monitoring and billing purposes. Smart metering would definitely affect the future development of the smart grid. In the near future, smart appliances would take advantage of the smart grid's functionality to synchronize their operations with its current state. For example, they may respond to pricing signals and decide when it is most economical to operate. An intermediate step before utilizing smart appliances could be the use of smart power outlets, for measuring the consumption of electrical appliances and controlling their operation in real-time.

Undoubtedly, these new technological advancements offer new possibilities for effective energy management and conservation in smart homes. Especially, when combined with the operations of the smart grid, these capabilities could offer great potential towards a coordinated, large-scale plan for energy efficiency.

Since current smart home solutions are vendor-specific and heterogeneous, and because this trend is expected to continue in the future, in order to enable the smart grid vision a common ground needs to be specified, i.e. a common language understood by all domestic premises, facilitating in-home and home-to-grid real-time interaction and communication. We anticipate that the Web is an appropriate solution for such a wide-scale interconnection (see Sections 1.2 and 1.3).

This chapter is organized as follows: Section 9.1 provides the architecture for the integration of energy-aware smart homes to the smart grid via the Web, emphasizing on the smart home environment. Through a proof of concept deployment and the emulation of a smart grid scenario,

[1] http://smartgrid.ieee.org/ieee-smart-grid

(near) real-time Web-based communication between the smart home and the grid is enabled, while some interesting applications related to the smart grid are investigated, from a smart home perspective. More specifically, in Section 9.2 the dynamic pricing program of the grid is exploited to schedule electricity-related tasks for a future time and in Section 9.3 load shedding is studied, as a technique to reduce total consumption for avoiding outages. Finally, Section 9.4 discusses other potential applications such as peak leveling and fault tolerance.

We note that the focus is on the ICT infrastructure that could be used for enabling flexible, reliable and efficient integration of smart homes to the smart grid. The proposed architecture targets the non-critical systems of the power grid. It is expected that the Supervisory Control and Data Acquisition (SCADA) system of the grid will be on a separate architecture/network. A SCADA system is an industrial control system used to monitor and control electrical power transmission and distribution.

9.1 Connecting Smart Homes to the Smart Grid

Smart homes become energy-aware by employing smart power outlets, for controlling the operation of electrical appliances in real-time. Our experimental smart home uses Ploggs (see Section 5.1.2) to achieve management of electrical appliances. Connecting energy-aware smart homes to the smart grid creates a new potential for saving energy and money.

The Web-based interaction between smart homes and the smart grid could be facilitated by utilizing intermediary devices called *smart grid controllers*. Each controller would be responsible for some houses or neighborhoods. More powerful grid controllers could manage larger areas such as villages and small towns. Smart grid controllers could maintain a hierarchical structure for fault tolerance and scalability. For example, controllers at higher levels (i.e. closer to the smart grid system) could be used for administering controllers at lower levels (i.e. closer to the customer

Table 9.1: Web API of a Web-based smart home for interaction with the smart grid.

No	Resource URL	REST Verb	MIME Type	Parameter (Type)
1.	HouseName/electricity	GET	JSON	-
2.	HouseName/reduceconsumption	POST	text/plain	reduction (Integer)
3.	HouseName/increaseconsumption	POST	text/plain	maxincrease (Integer)

premises). Low-level controllers would be able to communicate in a timely manner with each house while high-level controllers could interact with the main grid through a SCADA system. The overall system architecture is depicted in Figure 9.1.

We need to note that these smart grid controllers represent domestic controllers and should be separated from the main electricity infrastructure. It is also important to note that our focus is on integrating energy-aware smart homes to the smart grid, enabling flexible interaction patterns between them, and not on the operation of the smart grid and its controllers. The controllers could have specialized software for performing the grid's operations. An example implementation of a smart grid controller for emulating the behavior of the smart grid in a scenario involving load shedding is described in Section 9.3.

As mentioned in Section 4.2, the functionality of a smart home would be exposed as a Web API, by using the Web-based application framework for smart homes. Therefore, controllers can only interact with the house through the functions specified by this API. Vice versa, electric utilities could have their own Web API to inform the smart homes and smart appliances about current/projected tariffs. A simple Web API for Web-based smart homes, targeted to enable remote management and control by electric utilities is proposed in Table 9.1.

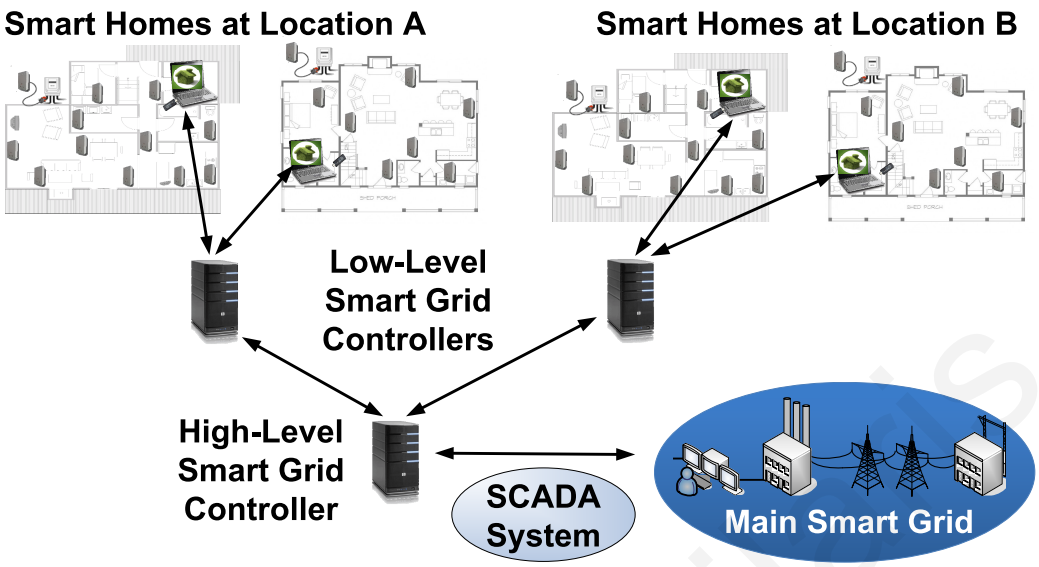

Figure 9.1: System architecture for integrating energy-aware smart homes to the smart grid through the Web.

This API would allow the utility to get informed about the total electrical consumption of the house (*GET electricity*), ask the smart home to reduce its consumption because an outage is possible (*POST reduceconsumption*) or allow the house to increase its consumption when the total load is in safe margins (*POST increaseconsumption*). The targeted quantity of reduced consumption is specified by the parameter *reduction* and it is defined in Watts. Similarly, the maximum allowed quantity of (increased) consumption is specified by the parameter *maxincrease* and it is also defined in Watts.

The responses from these HTTP requests are in standardized formats. The POST requests for reducing/increasing consumption are satisfied through a plain-text response, indicating a ACK/-NACK, while the GET request triggers a JSON response, including information such as consumption in kWh, instant consumption in Watt and a timestamp.

By following the physical mashup paradigm (see Section 4.2.6), home owners could automate their home very easily, combining electrical appliances with RESTful Web services offered by their electric utility. Through the Web, residents may pull easily the data they need from an open API offered by their electric utility, and use them right away in their own applications, in any programming language that supports HTTP. We list below a PHP script that implements a physical mashup in only six lines of code, which checks the current *hometariff* and starts charging automatically a plug-in hybrid electric vehicle (*PHEV*) as soon as the tariff falls below the defined *LOW_TARIFF* limit.

```
<?php
$tariff=http_get("UtilityAddress/hometariff/");


if($tariff <= LOW_TARIFF){
   $req=new HttpRequest("HomeAddress/PHEV/Switch/");
   $req->setOptions(array(state=>"ON"));
   $req->send();
   $response = $http_req->getResponseBody();
}
?>
```

Figure 9.2: Sample PHP code that checks the current energy tariff and charges automatically an electric vehicle.

We assume in this example that the utility exposes as a Web API information about its current tariffs. Offering real-time information is not infeasible for electric utilities. Recently, 3 utilities in California announced that they allowed their consumers to access their utility data through the Web. This initiative was called the Green Button [124].

It is crucial for the healthy operation of the smart grid to be assured that all HTTP calls to energy-aware smart homes would be executed reliably and on time. The application framework guarantees the successful execution of all Web requests by using a request queue for each smart power outlet and a fast retransmission mechanism (see Chapter 6).

To ensure the urgent execution of the requests that are made by the smart grid, the application framework supports also a priority mechanism (see Section 6.4.5). Hence, requests coming from the smart grid could be labeled as ”high-priority requests”, obtaining a prioritized execution, regardless of the current load at the request queues of the Ploggs smart power outlets.

Respecting the privacy of the consumers, the smart home may act as a ”black box”, allowing the smart grid to make requests to assure its proper operation but, at the same time, leaving the full control and responsibility on how to satisfy these requests to the residents, aiming to maintain a high comfort level at a reasonable expense. In such a way, people’s privacy and the healthy operation of the grid can be balanced.

The important question then becomes how to handle a request from the utility for reducing the overall electrical consumption. The most obvious approach would be to rely on the residents to assign priorities to their electrical devices. However, this may become inflexible and would complicate the whole procedure for the customers, who may have changing priorities and may not be willing to participate in such smart grid programs. An alternative approach would be to categorize devices according to their use patterns. Hence, we separate household devices into three broad categories. *Permanent devices*, which should never be turned off such as a fridge,

on-demand devices, which are utilized by home residents spontaneously, in order to accomplish a momentary task such as a toaster and *schedulable devices*, which are devices that are supposed to accomplish some specific task, but their operation is not momentarily urgent and can be postponed for a future time such as a dishwasher.

We focus mainly on schedulable devices since their operation can be postponed for low-demand and respectively low-tariff periods of the day. These appliances could be immediately turned off, in case there is a prompt call from the utility to reduce urgently the domestic energy consumption. Thus, customers are just required to identify which of their home devices are considered schedulable. Then, the application framework targets this device category to postpone use, in case there is a necessity. In the scenario when no schedulable devices consume energy and there is an urgent need for reduction, then on-demand devices would be selected.

It is trivial for the application framework to consider which devices are used on-demand, by observing their consumption patterns. Concerning permanent devices, the fridge is a special case as it could be turned off for some minutes without a problem. In addition, air conditioners and the electric heater could be special cases of on-demand devices whose operation may be postponed for some minutes.

In a complete smart grid scenario, different policies could have effect concerning the remote management of smart homes from the grid. Service-level agreements (SLA) could be used for assuring smooth supply of electricity and different customer pricing schemes could be applied. For example, ”gold customers” could pay a small extra fee for avoiding requests for possible reduction of electrical supply in peak periods.

9.2 Case Study: Exploiting Demand Response

A significant feature of the smart grid is demand response (DR). DR is about offering dynamic tariffs, according to supply conditions. Dynamic tariffs can be received in real-time, when utilities provide Web API to automatically disseminate them to the homes of their consumers. The DR capability would allow users to cut their energy bills by telling low-priority devices to harness energy only when it is cheapest. Since household appliances account for 50-90% of the residential consumption, their rational management is important for the efficient operation of the grid.

9.2.1 Implementation

The graphical Web application built on top of the application framework (see Section 4.3), includes also a task scheduling mechanism, for programming the execution of some pervasive service in a future time. We adapted this task scheduler, to support the DR feature from electric utilities. Hence, the residents are able to program actions to be executed automatically in low-tariff hours. A low tariff is specified as a lower percentage from the basic tariff, which is offered by the utility. As an example, the resident can program the electric water heater to heat water for a shower, when the tariff is 10% less than normally.

Furthermore, the resident is able to adjust the task scheduling procedure, according to his own preferences. He can define a maximum amount of waiting time, in case tariff does not fall below the specified limit in that time window. In this case, the task can start right after. The resident can also specify the execution of a task to be performed in morning, afternoon or night time. Finally, he can set the duration of each task, forcing the application to switch the corresponding electrical appliance off, as soon as the task completes.

To assess the feasibility of enabling energy-aware smart homes to the DR feature of the smart grid, we created a simulated scenario. We developed a Web server that simulates a (low-level)

smart grid controller, supporting DR functionality for EAC. A RESTful Web service was included, for informing the customers in real-time about the utility's current tariffs using RSS Web feeds. These tariffs, although simulated, aim to reflect the actual energy loads/demands of the country.

Figure 9.3 presents the total electricity demand in Cyprus, at a typical winter day. We assume that the power plants of EAC are able to operate in 4 different modes for generating electric power. These modes reflect the average electricity demands when dividing a winter day into morning, afternoon, early night and late night. These four modes can also be observed in the figure.

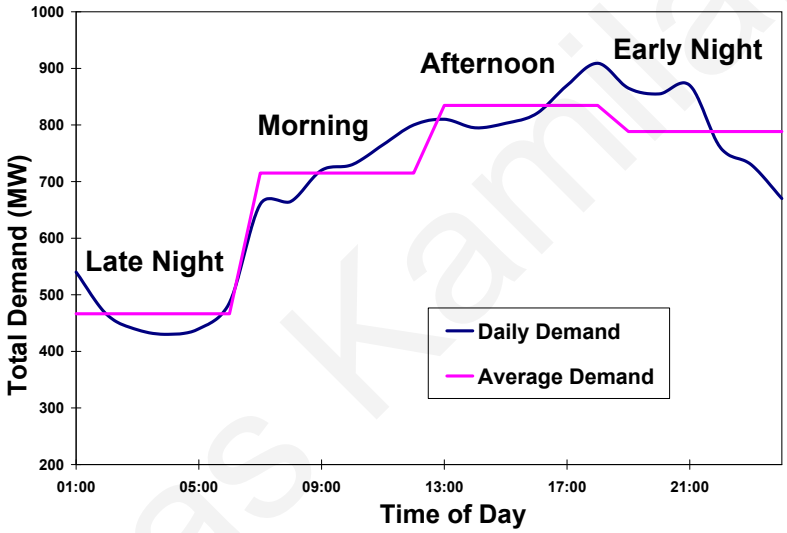

Figure 9.3: Total electricity demand at a typical winter day in Cyprus.

To produce dynamic tariffs, correlated to electricity demand patterns, this simple equation was used:

$$Tariff = \alpha \cdot BasicTariff \cdot (\frac{InstantDemand}{AverageDemand}) \quad (8)$$

where α is a coefficient used to weight the prices according to differences in demands. Using this equation with $\alpha = 1$, dynamic tariffs are produced that give incentives to consumers to utilize their electrical appliances in non-peak hours. These tariffs fluctuate around the basic home tariff, as shown in Figure 9.4.

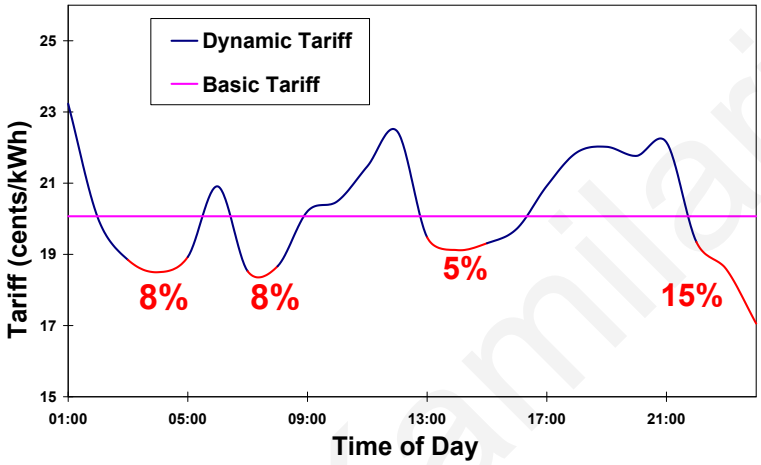

Figure 9.4: Real-time tariffs based on electricity demand.

9.2.2 Proof of Concept Deployment

Ploggs smart power outlets were used to create a wireless smart metering network inside an experimental smart home. Each Plogg was associated with some specific electrical appliance, in order to control its operation. The experimental setup is shown in Figure 9.5. In the figure, 5 hops are needed from the Plogg that acts as the base station, to reach the Plogg which monitors and controls the television.

A typical real-life scenario was considered. Most washing machines allow a user to define a preferred operation mode and start the washing in a future time. We programmed such a washing machine, through the task scheduling mechanism, to start the washing when the tariff is 5% less than its normal price. We tested the execution time of this task, by placing the washing machine and its corresponding Plogg in different hops from the application framework.

Figure 9.5: Experimental setup in demand response application.

Figure 9.6 illustrates the results of this experiment. The wireless network of the smart power outlets needs approximately 0.8-1.6 seconds to switch on the washing machine, which is considered an acceptable amount of time. Switching off the electrical device, after it has performed its task, requires around 1.2-2.7 seconds. This delay is caused by the firmware and the hardware

components of Ploggs. Since the task scheduling mechanism will operate for control scenarios with low workload, our results in regard to task execution times, are considered satisfactory.

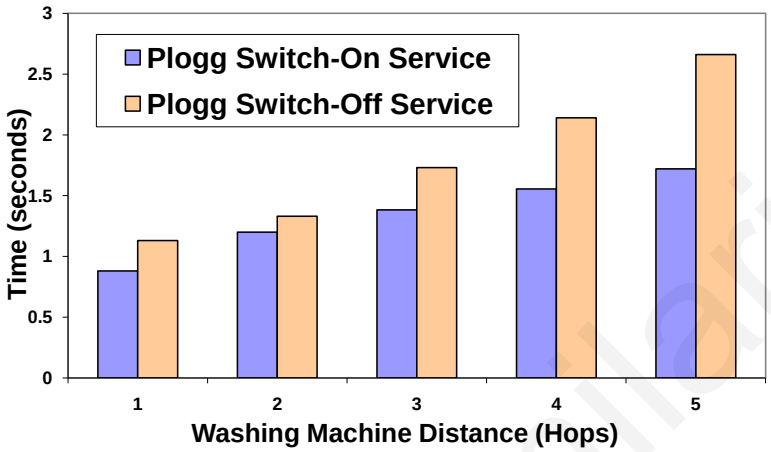

Figure 9.6: Task scheduling performance for switching on/off the washing machine.

9.2.3 Potential Savings

A new potential for saving energy and money is created by integrating energy-aware smart homes to the smart grid. We attempt to estimate this potential, considering that household appliances account for a significant percentage of the overall residential consumption.

As mentioned in Section 9.1, home devices span three categories: permanent, on-demand and schedulable devices. We are mostly interested in schedulable devices as they are devices that can be programmed to operate during low-tariff periods of the day. These devices can fully harness the DR feature of the smart grid. Therefore, electric utilities can save energy by better managing their load conditions and residents can save money by using electricity when it is cheapest.

Table 9.2: A list of schedulable electrical appliances and possible savings.

Device	**Consumption (kWh)**	**Duration**	**Month Freq.**	**Tariff Reduction in €**		
				10%	**20%**	**30%**
Hair Dryer	2.1 (1.5 - 2.5)	35 min	25	0,63	1,27	1,9
Electric Mixer	1.3 (0.8 - 1.5)	20 min	8	0,07	0,14	0,22
Electric Iron	2.8 (2.4 - 3.0)	2 h	8	0,93	1,85	2,78
Electric Oven	2.7 (2.5 - 3.0)	60 min	26	1,45	2,91	4,36
Water Heater	3.0	30 min	20	0,62	1,24	1,86
Dishwasher	1.5 (1.3 - 2.0)	40 min	22	0,46	0,91	1,37
Washing Machine	2.6 (2.0 - 3.0)	1 h 40 min	16	1,44	2,87	4,31
Clothes Dryer	3.6 (3.0 - 4.3)	50 minutes	11	0,68	1,37	2,05
Total Possible Savings				**6,28**	**12,56**	**18,84**
Electric Vehicle	3.3	8 hours	10	5,46	10,93	16,39

9.2.3.1 A Survey for Schedulable Devices

We performed a small-scale, telephone-based survey to identify schedulable electrical appliances. We discussed with 20 housewives and we posed to them the following question: *"if your electric utility offered cheaper tariffs at some hours of the day, which tasks, handled by your electrical appliances, would you schedule for these hours?"*. We also asked them about the duration of each task per day, translated into operational time of the appliance and also the monthly frequency, in which they perform this task. Their responses are listed in Table 9.2.

We used average values for task duration and monthly frequency. These values depend on many parameters such as weather conditions (e.g. electric water heater), number of family members (e.g. electric iron, washing machine), mentalities and habits of each society (e.g electric oven,

dishwasher) etc. Surely, these values are not absolute but just indicative, in order to facilitate our purpose of roughly estimating possible savings.

Observing the table, we can notice that some devices can be categorized both as schedulable and on demand (e.g. hair dryer, electric mixer, electric oven). Eight housewives commented that they were willing to use such devices according to low-tariff periods, in case this would save them significant money.

9.2.3.2 Calculations for Money Savings

After identifying home schedulable devices, we located some popular products for each device type and we recorded their energy consumption in kilowatt-hours (kWh). We computed the average values to derive typical consumption figures.

For our calculations, we used the home tariff offered by EAC in the period September-October 2012 (20,07 cent/kWh). In the last three columns of Table 9.2, we can view possible savings for schedulable devices, when home tariff falls 10%, 20% or even 30%.

Devices such as the electric mixer do not contribute in money savings and there is no need to schedule them for future time. However, devices such as the washing machine and the electric oven could contribute more effectively, allowing monthly savings of more than €1 each, when tariff falls 10% and more than €4, in case tariff falls 30%.

If we apply our optimized policy for all electrical appliances, monthly savings can be summed around €6 in 10% tariff reduction and up to €19 in case of 30% reduction. Considering the fact that the average monthly cost for electricity in houses around Cyprus is €175, possible saving of €19 gives 10,85% reduction in the bill of a typical home.

A survey from Parks Associates[2] remarks that *"over 80% of US households would pay up to $100 for cost-saving equipment if it chopped at least 10% off their monthly electricity bills"*. This survey indicates a possible acceptance of our approach by home residents. This is only valid under certain conditions such as effective tariff reductions by utilities.

An example killer application of the smart grid might be PHEV. It would be essential for the healthy operation of the grid to force residents to charge their PHEV in low-demand periods of the day (e.g. during the night). The last row of Table 9.2 shows possible savings when a typical PHEV exploits real-time pricing. We assume that a typical PHEV needs approximately 8 hours of charging in 3.3 kWh and provides driving range on batteries up to 150 miles[3]. Therefore, it would need charging every 3-4 days to adequately support the needs of a family. In this case, monthly savings can be more than € 10 when the tariff is reduced by 20%.

Since many schedulable devices are used only sometimes per month or some hours per day, it is not necessary to buy a smart power outlet for each different device. By purchasing 5-10 outlets, it may be enough to cover the daily needs of a family and schedule the operations of the devices through the task scheduling mechanism in low-tariff hours of the day. In this way, this approach would constitute a lower-cost investment.

Overall, DR is a promising capability of the smart grid. Energy-aware smart homes can exploit this functionality using the Web as an integration platform. This seamless interconnection has the potential for significant energy and money savings, both for the electric utilities and the customers, allowing Web-enabled electrical appliances to schedule their execution for low-demand and respectively low-tariff periods.

[2] http://www.parksassociates.com/

[3] http://www.hybridcars.com/electric-car

9.3 Case Study: Load Shedding

Load shedding [35] is an action taken to prevent frequency abnormal operation and is the last resort to maintain frequency stability[4], in case of contingency scenarios or autonomous-islanded operation. Such a scenario could include the non-scheduled outage of a generation unit or a main transformer. In this case, the non-served load previously served by the generator that currently experiences an outage will be allocated to other online units, given that their loading can be extended, remaining within the limits indicated by the manufacturer.

However, there are two unfortunate cases that might happen. First, the online generators may not be able to accommodate/undertake the extra load because they are already highly loaded. Secondly, online generation units might be able to accommodate the extra load (because they are not fully loaded) but, depending on the magnitude of the loss of generation, the response rate of their prime movers will not be in position to accommodate such a sudden increase in load, within the time slot indicated in transmission system regulations.

In such cases, to avoid an under-frequency abnormal operation of the power system, the operator is forced to apply a low-frequency demand control action, removing intentionally loads from service in order to prevent the total collapse of the system due to cascading events. This procedure is the definition of load shedding and lasts until the frequency magnitude recovers at the desired levels, when the rest of the online units are able to fully compensate the non-served load.

Load shedding is a procedure generally undertaken from the electric utility or the power system operator in a centralized way. The aim in this case study is to exploit the proposed architecture (see Section 9.1) and the functionalities provided by the Web API of Web-enabled energy-aware smart homes (see Table 9.1), in order to achieve selective load shedding that can be performed in

[4] Retain frequency within the operational and statutory limits.

a distributed manner, providing to the grid the capability of directly controlling domestic loads. This is a major characteristic of the future grid not yet standardized and its precise implementation has not yet been decided.

9.3.1 Implementation

The architecture described in Section 9.1 illustrates the path of the control messages and how the control objective will be achieved, i.e. the frequency stability of the power system. Concretely, the electric utility monitors the frequency of the power system almost in real time. In case of a critical variation based on the frequency value and its rate of change, control messages are issued from the utility control center to the high-level smart grid controllers, which order the low-level grid controllers to reduce the power consumption of the area that they are responsible for (e.g. a neighborhood). Then, the low-level controllers use the Web API of smart homes in a best-effort manner, asking the houses to reduce their consumption based on their current electricity demand and the condition of the grid.

Harnessing a smart home for these purposes has not yet been thoroughly explored by researchers and it is expected to revolutionize the future grid's structure and control. To demonstrate this potential application of the smart grid, an emulated scenario of selective load shedding has been implemented, employing three residential units in which the application framework for smart homes (see Section 4.2) has been deployed, along with 4-5 Ploggs at each house, associated with various electrical appliances of the house, mainly schedulable devices. The experimental setup is displayed in Figure 9.7.

In the scenario under consideration, it is assumed that the residential units are located in an islanded power system. A set of generators are committed and serve the total load of the system.

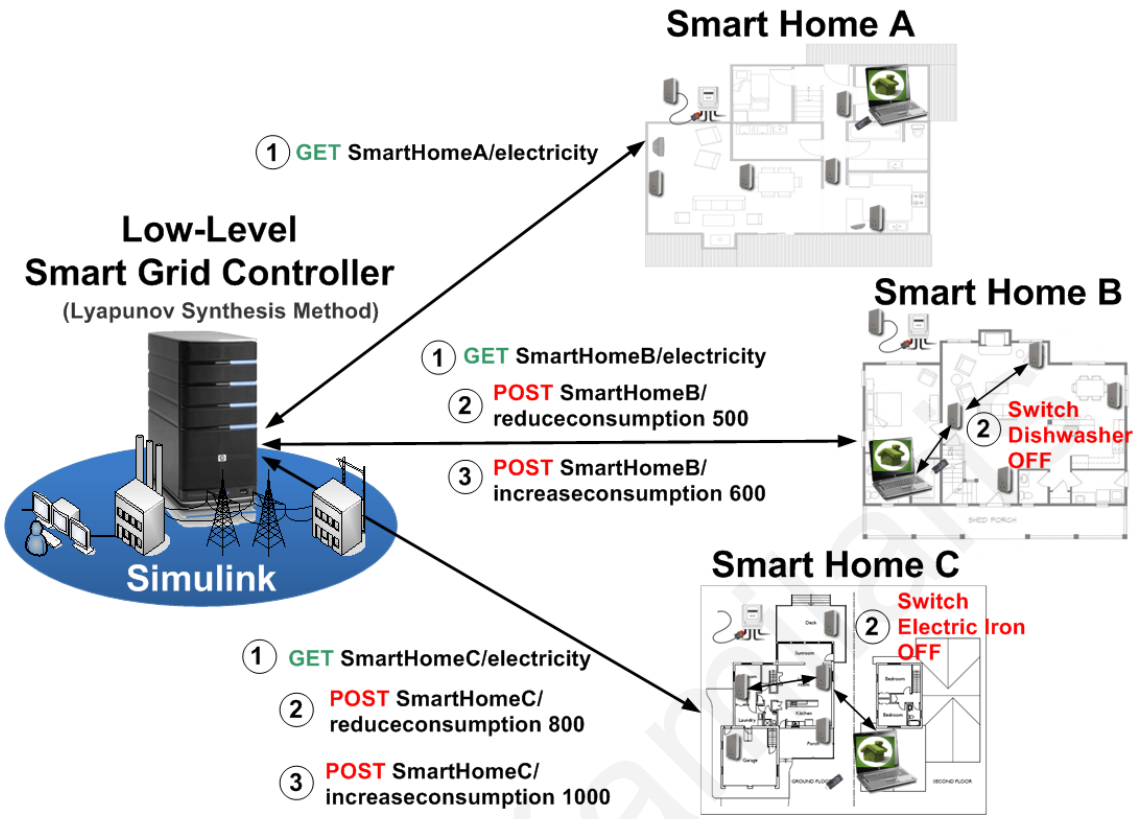

Figure 9.7: The experimental setup used in the scenario of load shedding.

Without loss of generality, the system is modeled with a generator, a transmission line and a variable load. The aim is to monitor the frequency stability of the system.

Virtual Phasor Measurement Units (PMUs) [133] are assumed, which are capable of measuring both the frequency and its first derivative online in real-time. System parameters were identified by applying the Lyapunov Synthesis Method [58], based on a simple linear second-order system frequency response (SFR) model [15]. Lyapunov Synthesis Method enables to identify the parameters of the plant through a suitable Lyapunov function, in terms of state variables and time, forcing this function to be at least negative semi-definite in order to obtain the desirable stability.

Further, a (low-level) smart grid controller was simulated on Simulink[5], and its main task was to maintain the stability of the system by determining the optimal (per unit) amount of electric load that should be shed to achieve frequency stability. The basic parameters of the simulation are listed in Table 9.3.

Table 9.3: Parameters at the simulation of a smart grid controller.

Parameter	**Value**
Simulator Time Step	0.27424 msec
Total Simulation time	150 sec
Sampling Rate for Domestic Consumption	350 msec
Total Number of HTTP requests	428

In the scenario under discussion, the grid experiences a sudden, non-scheduled increase in load. This increase has been modeled as a step function change in demand. As a result, the frequency of the power system starts to decline continuously. Three different cases are considered:

1. No load shedding takes place.

2. Load shedding is performed following the conventional practice that is applied by the vast majority of power system operators worldwide.

3. An intelligent selective/soft load shedding is performed, exploiting the proposed system architecture.

In the first case, the frequency decline is just monitored without taking any action. In the second case, circuit breakers are activated, shedding load based on an approximate rule of thumb, which indicates that the connected load magnitude should be decreased linearly in relation to the

[5]http://www.mathworks.com/products/simulink/

frequency decline. For example, if the frequency decreases by 1%, then load magnitude would be decreased by 2% [14]. Thus, when frequency declines 1% becoming 49.5 Hz, 2% of the load is shed. At the frequency level of 49Hz, a 4% of the initial load is shed.

Finally, in the third case, the proposed algorithm was applied to determine when load shedding should be performed and the amount of load that should be shed. Current consumption is monitored in near real-time by the operator of the grid through the relevant smart grid controller, which is responsible to aggregate the consumption of the houses it controls. The grid controller obtains current consumption by issuing an HTTP *GET electricity* command to each smart home, in regular time intervals of 100 ms. The total consumption, input to the simulated power system as a load, has been derived from the current consumption of the three domestic units, multiplied by a factor of 10,000. In this way, the emulated scenario has been scaled to an islanded grid.

Based on the total consumption, the current value of the grid's electrical frequency and its rate of change, the operator asks from the grid controller to shed the required amount of load. Then, the controller asks from each house to shed a specific amount of load by issuing an HTTP *POST reduceconsumption* command. The targeted amount of load to be shed at each home depends on its current consumption. After that, the application framework decides which device(s) should be switched off based on the policy employed (e.g. according to device category and its current consumption). In this implementation, schedulable devices whose consumptions were closest to the targeted reduction were preferred to be switched off. This is an iterative conversational procedure that takes place until the targeted total amount of load is shed. As soon as the command is successfully executed (an ACK has been received and the corresponding devices have been switched off), the simulator is fed with the scaled amount of shed load.

Finally, when the grid is in a "safe" condition again (the frequency has recovered in normal/desired limits), the controller starts progressively to issue HTTP *POST increaseconsumption*

commands to the smart homes, to gradually add the load that has been previously curtailed, allowing a maximum restoration in load at each house. Hence, the application framework switches on the schedulable devices that had been previously switched off, allowing them to finish their task.

The three different phases of the proposed algorithm can be observed in Figure 9.7. In this specific scenario, the grid controller monitors the electrical consumption of all the three houses and, at some time, needs to perform load shedding in order to maintain frequency stability of the system. Hence, it decides to issue HTTP POST commands to smart homes B and C to reduce their instant consumption by 500 and 800 Watts respectively. Smart Home B responds to this command by switching off the dishwasher and smart home C by switching off the electric iron. These are schedulable devices whose operation may be postponed for a future time. Then, when the system is stabilized again, the grid controller issues another POST command to smart homes B and C, allowing them to increase their consumption by 600 and 1000 Watts respectively.

9.3.2 Evaluation

Figure 9.8 depicts the performance of the three different schemes in regard to the time response of the frequency variation. When no load shedding is applied, the frequency declines exceeding the permitted limits (of +/- 3 Hz), causing under-frequency abnormal operation. In this case, the power grid will experience instability and cascading events will possibly follow, causing a total blackout. Furthermore, devices such as induction motors can be damaged or even burned out at low/under-frequency operation.

In the second case, when conventional practices regarding load shedding are performed, the frequency exceeds the desired levels for four seconds and needs 35 seconds in total in order to ”absorb” the disturbance. After this critical time, frequency oscillations still exist, being reduced

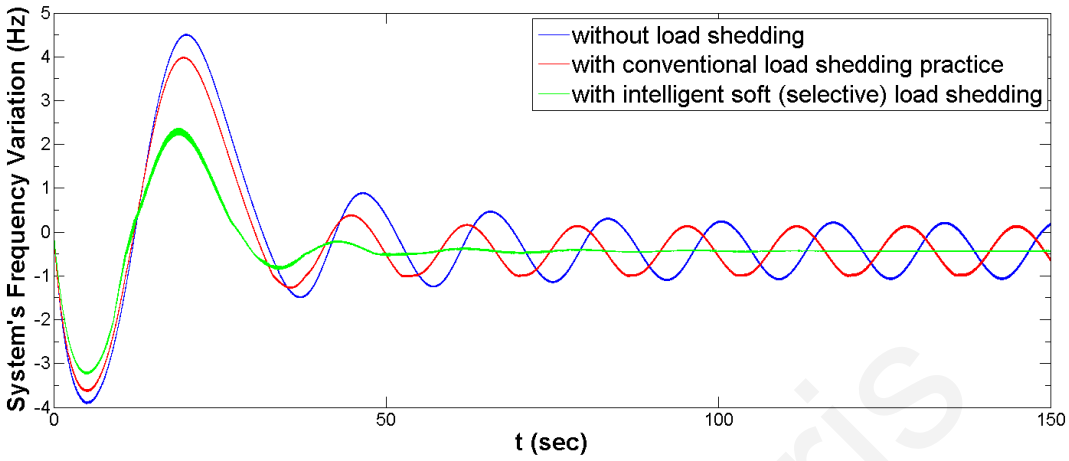

Figure 9.8: Load shedding using different practices.

with a small step. In this case, a set of customers would experience a total outage causing major discomfort to them.

Finally, when our intelligent selective load shedding algorithm is applied, the frequency remains at the desired levels during the whole emulation time. It recovers fast in the first 35 seconds of the emulation and totally absorbs the disturbance after 50 seconds.

Moreover, because the intelligent algorithm is sensitive even at small frequency variations and acts proactively, based on the frequency rate of change, it finally achieves a minimum total amount of load to be shed. This is in contrast to the conventional practice in which a larger amount of load needs to be shed in order to achieve the same control objective, i.e. to maintain the system frequency in the desired levels. In addition, load restoration can be applied smoothly as soon as the frequency experiences an upward trend, exceeding a certain threshold with certain momentum (certain rise rate given by the positive time derivative of the frequency). In this way, the maximum load is restored in minimal time.

Our findings indicate that the load shedding application contributes to a more robust power grid that controls frequency oscillations in a more effective and efficient way, preventing power system instabilities and total outages of utility services. At the same time, it minimizes unexpected outages and customer discomfort by minimizing the load to be shed and by restoring it faster than the conventional practices do.

9.4 Other Potential Applications

In the following subsections, some other potential applications that could be enabled using the proposed architecture are briefly described. These applications include peak leveling, fault tolerance, billing and a distributed electricity market.

Peak Leveling/Shaving

Peak leveling/shaving is a process that aims to eliminate the demand in peak hours and to shift it in non-peak demand periods. In this way, the demand curve is leveled, providing maximum exploitation of the current utility infrastructure while the need for excessive spinning reserves is reduced. For example, in the case of an expected peak hour (e.g. a world cup final), expensive generators are supposed to be employed that are able to switch on fast and follow the load changes quickly. However, this is costly for the utility, both in operations and expenses.

The proposed end-to-end architecture (see Section 9.1) aims to mitigate this situation by applying peak shaving, shifting the low-priority and reschedulable domestic loads to non-peak demand hours, in the way of distributing evenly the produced energy throughout the day.

The approach described in Section 9.3 for load shedding could also be employed for peak shaving. In addition, demand response programs could be utilized (see Section 9.2), using special tariffs to influence consumer behavior.

Fault Tolerance

An important characteristic of the smart grid is the timely detection and localization of faults. The proposed architecture (see Section 9.1) has been designed in order to facilitate this task through the hierarchical, distributed structure of the controllers.

The established end-to-end connections between the smart grid controller and each of the smart home application frameworks provide in (near) real-time the customers' consumption. In addition, the HTTP POST requests issued by the controller, destined to the customer premises, reveal information about the status of the house, for example if a command has been successfully executed concerning shedding some amount of load.

In this way, the controllers can identify any abnormal behavior of the customers' consumption patterns and find out quickly which customers experience malfunctions or outages. Further, if the smart home framework does not respond to the issued commands multiple times (the customer-specific commands are re-sent until an ACK is received), this implies that the specific user experiences a failure or violates the SLA contracted with the operator of the grid.

In such cases, the smart grid controller may send some additional requests to the home framework in order to get informed about the status of the house and increase its awareness about the situation. Hence, the customers who experience unavailability of some or all services will be automatically detected and will be associated with the faulty feeders. Then, the utility crews will locate the fault and rush in immediately, reducing the duration of service unavailability (outage).

Their fast response would increase the reliability index of the utility company and generally improve its severity-based indices. Indices including expected unserved demand per year and expected unserved energy per year would be improved because of the faster faults restoration and the early actions taken as a result of the enhanced situational awareness of the grid.

Of course, there exist situations that could complicate fault localization. For example, a massive DoS attack on the customer premises and the smart grid controllers could hinder any efforts for fault identification and solving. Security countermeasures such as firewalls could be employed to prevent such attacks. A small discussion about security is performed in Section 10.5. Nonetheless, since the proposed architecture relies on the Internet/Web, fault tolerance in general can not be guaranteed.

Billing

Billing is a major source of expenses by electric utilities since dedicated personnel must be employed for reading the home meters manually, requiring the physical presence of an employee at each house. The smart grid can provide long-term savings to the utility by providing automated meter reading via frequent interactions with energy-aware smart homes, through the Web.

Considering the Web API offered by smart homes as provided in Table 9.1, it could be enough for the utility to issue GET requests in frequent intervals (in magnitude of seconds/minutes), to get informed about the instant domestic consumption. However, this pull-based technique would not scale for millions of houses. Therefore, a push messaging technique based on a publish/subscribe model (see Section 2.6), similar to the one used in Section 5.2.4, would be more appropriate.

Availability of timely billing information would help residents to become more aware about their electricity footprint, giving them financial incentives to reduce their consumption.

A Market for Generation/Consumption of Electricity

The use of domestic renewable electricity generators could become a trend in the future. These generators would form decentralized energetic islands or *microgrids*. A microgrid consists of many small, distributed energy resources, located near each other in the low-voltage distribution system, connected to each other through some network.

Through this decentralized network architecture, smart homes may be capable of trading energy for money by means of market agents. These agents would represent electric devices, either generators (e.g. photovoltaics, wind turbines), or loads (e.g. television, fridge).

Hence, real-time auctions about electricity could be developed among energy-aware smart homes and the smart grid. According to the current generation and demand, smart homes would sell/buy electrical energy in competing prices.

The proposed architecture defined in Section 9.1 could be employed for enabling a real-time market for energy. The Web could become the platform for handling the message exchange between houses and the grid. More specifically, the Web API offered by smart homes, as shown in Table 9.1, could be extended to support also market-related functionalities. For example, assuming that a smart home needs energy in some sunny day, its home market agent could query another smart home that generates electricity from photovoltaics, asking about the price of produced energy. This could be achieved by issuing a *GET energyprice* request. In case a deal is accomplished, this home agent could issue a *POST energydemand* command declaring the needed amount of power. Of course, this is only a simple example, since a complete solution would require money transactions between the two parties, intelligent algorithms for automating the purchase of energy in competing prices and an infrastructure for offering the distributed generated electricity to other houses.

Chapter 10

Beyond the Smart Home Environment

This chapter presents challenges encountered in the research area of the WoT and are beyond the smart home environment. These challenges relate to Web-enabled smart homes and their functionalities in a significant degree, however, their application scenarios involve general and larger-scale pervasive spaces, urban environments and global discovery of environmental services. In all these scenarios, the research performed in this thesis about enabling smart homes to the Web may be directly applied, to achieve the overall goals defined at each scenario. The smart home constitutes a vital living cell in the ecosystems envisioned by these challenges. Also, in real deployments within the home and beyond, issues of security and privacy need to be addressed.

The rest of the chapter is organized as follows: Section 10.1 discusses the extension of the application framework for smart homes in larger-scale, mobile, ad hoc ubiquitous scenarios and Section 10.2 explores augmenting other real-life pervasive applications, beyond smart homes, with social characteristics. After, Section 10.3 aims to address a crucial issue at the WoT agenda, which covers the global, real-time discovery of physical devices and environmental services through the Internet/Web. Then, Section 10.4 explains how advanced knowledge inference could be achieved

in urban environments, helping the citizens to enhance the quality of their everyday lives. Finally, Section 10.5 considers some security aspects inside Web-based smart homes and beyond.

10.1 From Smart Homes to Smart Spaces

Probably the reader has already wondered whether the application framework for smart homes (see Section 4.2), could be generalized for other pervasive scenarios, indoor or outdoor. Of course, each different pervasive scenario includes specific requirements and has particular characteristics. For example, some pervasive physical spaces might be characterized by high unreliability, increased percentages of transmission and device failures, high mobility of sensors and actuators etc. The interaction between the users and the embedded devices of the pervasive space might involve any type of interaction (e.g. request/response, event-based, streaming).

The application framework has been designed with its primary focus being on indoor, home environments. However, the framework is flexible enough to adapt to many different application scenarios successfully, offering satisfactory performance. The requirements defined for its development (see Section 4.1) are general enough, to allow the framework to be easily extendable to support various real-life applications. A direct, straightforward application of the application framework could be smart buildings with numerous tenants, visitors and workers. A real-life application of the framework in an organic farm is presented in Section 10.2, in which the workers of the farm monitor a farm's greenhouse through the application framework. Since the framework supports more than a hundred simultaneous users (assuming they are making 1 request per minute to their home environment, see Sections 6.4.1 and 7.2), it could be well applied in scenarios involving building automation.

Some important preferences and features of the application framework that facilitate its general applicability in ubiquitous spaces are:

- Robustness by identifying device failures fast through aliveness checks and masking these failures by forwarding requests to relevant physical devices offering the same service (see Section 4.2.4).
- Reliability by masking transmission failures to/from the embedded devices, supporting fast retransmissions (see Section 6.4.2).
- Interoperability between heterogeneous physical devices and their services by following REST architectural style (see Section 1.3).
- Various interaction possibilities with physical devices in a uniform manner (ad hoc interaction, continuous monitoring and event-based messaging).
- Support for multiple users who may interact with the application framework simultaneously (see Section 6.4.1).
- Graphical user interfaces for facilitating the management of the physical environment by end users (see Section 4.3).
- Direct access for users to their physical space pervasively through the Web.
- Promoting application development by end users with little programming experience, through the use of physical and urban mashups (see Sections 4.2.6 and 4.2.7).
- Load balancing for better serving high traffic (see Section 6.4.4).
- Support of prioritized requests, necessary in time-sensitive scenarios (see Section 6.4.5).
- Support of a caching mechanism, to reduce the burden of embedded devices (see Sections 4.2.5 and 7.2).

- Acceptable performance in terms of request/response times, energy performance of physical devices and push times (see Chapter 7).

Even though the previously listed features may promote the use of the application framework in various pervasive scenarios, there are also some issues that need to be considered and addressed before proceeding with such an initiative. At first, our evaluation efforts have focused on test cases with arrival rates of 2 requests per second or at most 120 requests per minute. These test cases cover typical smart homes and buildings hosting around tens or a hundred of home members (assuming that all these members are active users of the relevant smart home application).

For larger-scale pervasive cases, especially outdoor, in which many hundreds or even millions of users might need to be supported, the application framework may not be suitable. This estimation is reinforced by observing the push time performance for forwarding streaming data to interested subscribers (see Section 7.4). Push times grew exponentially when more than 60 subscribers were involved. Of course, this statement is valid considering that the framework has been installed only on a laptop computer during the evaluation efforts (see Section 7.1). Anyway, more focused evaluation using better physical equipment must be performed, in order to determine the scalability of the system in terms of user numbers.

Scalability issues concern also the supported number of physical devices. Our evaluation efforts involved at most a multi-hop 2-3-4 topology with 9 sensor motes in total. Larger deployments need to consider also larger numbers of sensor and actuator devices, as well as various topologies and hop-distances of devices. As we mentioned during our evaluation attempts, we selected a small number of sensor devices to stress test the system in high traffic and demanding conditions. Our findings indicate that the application framework and the request queue mechanism behave well in these conditions. Hence, it is mainly a matter of the 6LoWPAN protocol (see Section

2.1.6) to support wireless, multi-hop networks with hundreds of embedded devices. A small experiment performed in Section 4.2.2, indicated the capability of the framework to support a few hundreds of devices, dependant on the computing device type on which it has been installed.

An important issue that may be encountered in general pervasive spaces is the possible mobility of the embedded devices. The design of the request queue mechanism (see Section 6.2), as well as our analysis and evaluation procedures assumed a static deployment of home devices inside the house ecosystem. Also the equation for setting the request queue retransmission interval (see Section 6.3.2) was selected based on a static placement of embedded devices. Merely a general discussion about the support of moving objects and dynamic environments was provided in Section 6.3.3, however, no further analysis was performed. Hence, mobility was not adequately addressed in this thesis and might be an important requirement in specific pervasive scenarios.

To sum up, our application framework for smart homes offers numerous features and benefits that enable its applicability in various pervasive scenarios. However, the framework is not suitable to be employed in scenarios demanding high scalability, mainly in terms of users and in which high mobility of physical devices is present. More detailed and analytical evaluation efforts need to be performed to assess the ability of the application framework to scale to hundreds or millions of users and physical devices.

10.2 Online Social Networking of the Physical World

The case studies performed in Chapter 8, indicate that social influence is important in influencing people to save energy and to engage in sustainable lifestyles.

These studies suggest that online social networking platforms have the potential to support pervasive applications that target the physical world, enhancing these applications with a social

shape. Physical devices and environmental services can become ubiquitous inside social networking sites, merged with the everyday social activities of users.

To further demonstrate this concept, we developed a case study in another real-life scenario that focuses on environmental monitoring, beyond the smart home environment. RiverLand Dairy Farm is the first organic farm in Cyprus, involving biological procedures to produce animal and vegetable food products. The farm has numerous greenhouses. Monitoring the environmental conditions inside the greenhouses is crucial for the proper growth of vegetables.

10.2.1 Implementation

We deployed the equipment described in Section 5.1, namely Telosb sensor motes and Plogg smart power outlets in one of the farm's greenhouses, in which tomatoes are grown. The intention was to monitor through Facebook in real-time the temperature, humidity and illumination levels at the greenhouse as well as the electricity footprint of a lamp that is placed inside it.

Social Farm Facebook application enabled the monitoring of the conditions at the greenhouse by the workers of the farm, through Facebook. Social Farm and our application framework were installed on a laptop that was also placed inside the greenhouse.

The farm employed eleven workers and we asked them to participate in our experimental deployment by utilizing for two weeks the Social Farm application. Two workers, who were quite old, kindly refused to participate as they had zero experience with computers. From the rest of the nine workers who accepted to participate, five already owned a Facebook account and four of them had never used a SNS before.

We harnessed Facebook groups as the mechanism to achieve access control. We created the group *FarmWorkers*, which operated privately. Only the farm owner could approve join requests.

We also included the eventing infrastructure described in Section 8.1.1, to allow workers to get notified when something abnormal happened at the greenhouse.

10.2.2 Evaluation

After the two weeks, we asked from the workers to express their impressions in using our application. The methodology used for the evaluation procedure was elementary, based on small chats and personal interviews.

All of them found the application easy to be used. They agreed that the integration of physical devices in SNS would promote sharing of sensory services. A worker even suggested a Facebook application that targets health monitoring, where doctors would interact with their patients monitoring, at the same time, the patients vital signs.

The workers who already owned a Facebook account were excited with the perspective of controlling the greenhouse while amusing with their friends. The workers who did not have previous experience in Facebook found it difficult to understand the notification methods and were not so enthusiastic with this approach. Some complained that the notification methods are not very effective since the user must be online to be informed. Nonetheless, all the workers stated that Social Farm increased their monitoring activity, making them more aware about the farm in general.

The owner of the farm, although he was one of the workers who did not previously possess an account, was really satisfied with the application and showed interest to learn the costs needed to fully automate his farm. He was excited with the possibility of delegating access, through the FarmWorkers group to his friends/employees, to observe the status of his greenhouse.

Summing up this small study, the enhancement of pervasive real-life applications with social elements seems to help people to engage in beneficial activities, such as monitoring a greenhouse

for abnormalities. Blending working tasks with online social entertainment may give some incentives to people, for adopting sustainable lifestyles.

10.3 Global Discovery of Environmental Services

In the future, Web-enabled sensor devices (see Section 5) are expected to constitute the large majority of the Web population, and will be deployed around the world to measure with high accuracy the physical environment. These devices will mainly be exposed to the Web by means of Web-enabled smart homes, which would offer their functionalities as open Web API.

In general, there do not exist yet standardized, scalable and flexible ways to globally discover Web-enabled embedded devices, based on their characteristics and capabilities. In other words, a real-time search engine for the physical world is still unavailable. Dyser [125] and Snoogle [172] are early efforts towards real-time discovery of physical entities, however, they either require additional Web infrastructure or they do not scale for the Web. On the other hand, microformats[1] suggest ways for making HTTP data available for indexing and searching, but their utilization for global discovery of the WoT would augment our dependencies to commercial search engines.

We believe that service discovery of embedded sensor devices needs to be ubiquitous to the users of the Web. The proposed solution must comply with existing Internet standards and should not require major changes to the existing technical equipment and protocols. Users should be able to discover environmental services simply by typing related keywords in their favorite Web browser. In this way, discovery of physical devices may be similar to the way we discover Web sites through search engines. In the following subsection, we present an approach for performing real-time discovery of the WoT by using the existing Internet infrastructure.

[1] http://microformats.org/

10.3.1 Case Study: DNS-based Real-Time Discovery

We propose to exploit the existing Internet infrastructure to achieve real-time discovery of physical devices and environmental services. We investigate utilizing the Domain Name System (DNS) as a scalable, pervasive, global meta-data repository for embedded devices, and its extension for supporting location-based discovery of Web-enabled physical entities. DNS is a hierarchical, distributed naming system for computers, whose main functionality is the translation of domain names meaningful to humans into IP addresses meaningful to machines. Applying an existing technology for the discovery of physical devices and services, especially one that has been ubiquitous for decades, offers many advantages, including well-defined support, easy configuration, experienced developers and users, as well as availability of open-source implementations.

The general idea of exploiting DNS for service discovery is not new. DNS-based Service Discovery (DNS-SD) [28] proposes using standard DNS programming interfaces, servers and packet formats to browse a network for services. Similarly, Multicast DNS (mDNS) [29] enables to perform DNS-like operations on a local network in the absence of any conventional unicast DNS server. Even though both protocols have been originally designed for device/service discovery in local networks, DNS-SD has been extended to provide wide-area service discovery. However, this functionality for service advertising is domain-centric, meaning that users are only able to be informed about services offered in some particular domain. In contrast, our proposal is service-centric, envisioning to enable global, real-time, location-based discovery of pervasive services, offered by Web-enabled sensor devices. In the following subsections, we examine how to change the organization of DNS to support location-based, real-time discovery of environmental services.

10.3.1.1 Device Registration

We propose the inclusion of a new top-level domain at the DNS, which is the *env* domain. This domain is intended to support all embedded devices and environmental services which are enabled to the Web and registered to the DNS. Specialized authoritative name servers may be responsible for this domain. These .env domain name servers shall allow real-time registration of physical devices and their (RESTful) Web services through Dynamic DNS (DDNS) [139].

In general, a particular service[2] instance can be described in DNS using SRV [77] and TXT [118] records. A SRV record has a name of the form:

$$< Instance > . < Service > . < Domain > \tag{9}$$

and gives the target host and port where the service instance can be reached. The DNS TXT record of the same name gives additional information about this instance, in a structured form using key/value pairs

Whenever a Web-enabled sensor device becomes available on the Web, it would create a request to the .env DNS server, asking for registration. In this request, the device must specify its name, location and the services it offers. The .env server could offer a Web API, allowing sensor devices to POST their discovery details in HTTP requests. In case all information is provided, the .env DNS server would acknowledge the request, assigning a fully qualified domain name to the device, registering it in a SRV record.

Hence, each sensor device will be assigned a unique hostname that has the following format:

$$devicename.\_http.\_tcp.service.location.env \tag{10}$$

[2]To avoid confusion, we note that by the notion of a "service", the DNS refers to the Internet protocols used to interact with some domain. In the WoT (and in this thesis), services are meant as the Web-based functionalities offered by Web-enabled sensor devices.

Table 10.1: DNS record translation.

No	Record	Explanation	Example Record/URL
1.	SRV	A device instance offering *service* at *location*	device._http._tcp.pollution.nicosia.env
2.	PTR	Lists devices supporting *service* at *location*	_http._tcp.pollution.nicosia.env
3.	A	Translates a URL to a 32-bit IPv4 address	device.pollution.nicosia.env
4.	AAAA	Translates a URL to a 128-bit IPv6 address	device.pollution.nicosia.env
5.	TXT	Meta-data about the selected sensor device	device._http._tcp.pollution.nicosia.env

where *devicename* is the user-friendly name of the sensor device (<Instance> portion of the SRV record), *service* is the actual Web service offered by it and *location* is the absolute location of the device. Service and location define the <Domain> portion, which becomes *service.location.env*. The <Service> section is either *_http._tcp* or *_http._udp*.

A sensor device may offer various environmental services. Thus, multiple hostnames will be created for this device, each for a different service. Since the WoT proposes a resource-oriented architecture, services would be described with uniform resource identifiers (URIs) (see Section 5.2.3). This avoids ambiguities when users construct environmental URLs in their Web browsers.

10.3.1.2 Service Discovery

Service discovery begins when a user types in his Web browser a URL ending with the .env label. Since the user may not be able to construct a URL similar to the one in (10), he could ask about all sensor devices offering *service* and deployed in *location*. This would be achieved by using the PTR DNS record type [118], as shown in Table 10.1, by specifying a query of the form <Service>.<Domain>. In this case, DNS would respond to the PTR lookup with a list of relevant <Instance> records.

In case numerous relevant records exist, a technique such as round-robin DNS could be employed. Round-robin DNS is a technique for achieving load distribution and load balancing by responding to requests from various clients according to an appropriate statistical model.

After the user receives the list of relevant instances, he can use any of them to construct a query similar to the one shown in (10). By typing this URL in his Web browser, the DNS will translate it to the actual IPv4/IPv6 address of the corresponding Web-enabled sensor device. This translation is realized by using the A and AAAA DNS record types, as shown in Table 10.1. The user can also use the TXT DNS record type to receive general characteristics/capabilities of some sensor device, in case they are available.

When the IP address is resolved and the request is forwarded to the appropriate sensor device, the device needs to respond with a description of its functionality. This is necessary in order to understand the semantics of the device (e.g. indoor/outdoor, degree of accuracy, measurement unit) and how to interact with it to get informed about the environmental conditions. Thus, a description language must be defined, which declares the device semantics, but also explains the interaction possibilities with it.

Example description languages that could be adopted are Extended Environments Markup Language (EEML)[3] and SensorML[4]. These are languages that describe the capabilities of sensor devices, but do not describe effectively the interaction possibilities with them.

Thus, we propose to employ WADL (see Section 5.2.2), which is a platform- and language-independent way of describing HTTP-based applications and services. After the user receives the WADL description, he can construct the appropriate request, query the sensor device and get informed about the environmental conditions in the selected location, in a well-known format such

[3] http://www.eeml.org/

[4] http://www.opengeospatial.org/standards/sensorml

as XML or JSON. The whole device registration and device discovery procedure is provided in Figure 10.1.

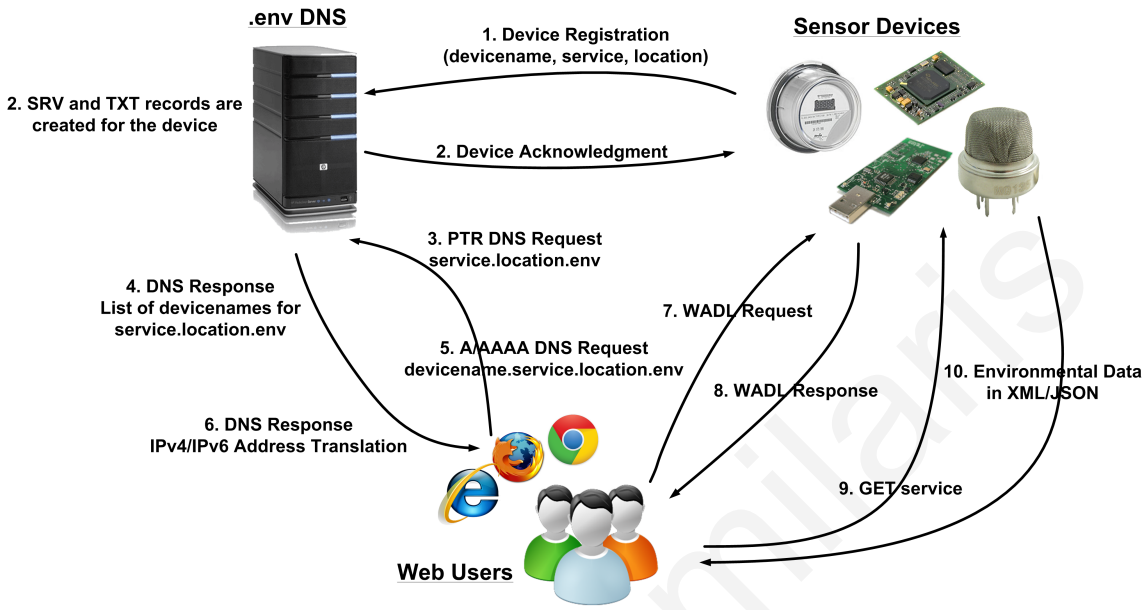

Figure 10.1: Environmental service discovery procedure using DNS.

Environmental service discovery may involve not only humans but also machines, for automated M2M communication. In this case, machines could discover useful services on demand, and take decisions automatically based on current environmental conditions.

10.3.1.3 Freshness of Information

The idea of online directories for Web services (e.g. UDDI[5] for WS-* Web services [13]) never worked, mainly because of information inconsistency and unavailability. Through DNS, these inconsistencies can be avoided.

[5] http://uddi.org/pubs/uddi_v3.htm

In general, the DDNS service allows the DDNS server to allocate a static hostname to a Web-enabled physical device, and whenever the device is allocated a new IP address, this is communicated to the DDNS provider by software running on the device.

Furthermore, Dynamic DNS Update Leases [30] constitutes a method of extending DDNS to contain an update lease life, allowing a DNS server to perform DNS dynamic updates with an attached lease time, that are automatically deleted unless renewed before the lease expires.

Hence, the name server ”forgets” after some time interval the registration of the sensor devices. Devices may be forced to state frequently their operability to the DNS server, by re-registering, e.g. every some hours. In this way, dynamic IP address assignment to devices could be supported and also device unavailability could be identified in a relatively small delay.

10.3.1.4 Practical Issues

The proposed approach creates numerous practical issues that must be effectively addressed. The general operation of the DNS is expected to be affected, mainly in terms of management, security and increased traffic. Concerning management issues, since the DNS follows a hierarchical structure, the addition of the .env top-level domain is not expected to be a complicated task. Availability of devices and services may be assured by DDNS update leases.

Partial reliability could be ensured by checking the IP addresses of the sensor devices, if they fall in the locations claimed by them during the registration process. More reliability and trust would be achieved if some user-based feedback system was applied for sensor devices and their environmental services, similar to the way eBay works for rating its users.

User-based rating could be realized if sensor devices had a social presence on the Web. Towards this direction, the company Evrythng[6] aims to create Web-based social profiles for physical

[6] http://evrythng.net/

devices. It would then be easy for users to rate them, according to their quality of service. Such an initiative would also provide more advanced privacy, allowing the owners of the devices to share them only with their family, friends or everyone. In general, the social Web could be harnessed for authenticating and authorizing users to interact with environmental services. We have already developed such an application prototype for device/service sharing in Section 8.1.

Moreover, this approach requires unique device name assignment to sensor devices that offer the same services and share the same location. Upon a conflict, the DNS server should automatically select a new name for the device, typically by appending a digit at the end of its name. Since the .env DNS server would constitute a distributed repository, the devicename assignment should be visible to all .env name servers. Due to the massive amount of Web-enabled devices in the near future, this could be the cause of high load to DNS. However, this issue may be mitigated by area- or location-based assignment of devices to the .env DNS servers.

Another important issue concerns naming of services and locations. For example, locations are named differently in each language (e.g. ”Lisbon” in English vs ”Lisboa” in Portuguese). To avoid these ambiguities and guarantee uniqueness, electronic directory services such as X.500[7] could be used, where a distributed database would contain unique translations for services/locations.

Since the DNS system supports a built-in caching mechanism, this mechanism could be exploited to support caching of sensory data. During their registration, sensor devices would include their latest measurements, stored in TXT resource records. These measurements could then be forwarded to users who query the .env DNS server for a relevant service, while they are still fresh[8]. Nonetheless, current DNS infrastructure can not tolerate such high loads.

[7] http://www.x500standard.com/

[8] Defining the freshness of measurements varies between devices and services.

Finally, the whole system would be optimized by forcing the .env DNS server to return directly the IP addresses of relevant sensor devices and not their hostnames. However, this may not be practical since the IP addresses of the devices may change frequently while their hostnames remain the same. Furthermore, some users might prefer to use some particular sensor devices they found more reliable than others. By exchanging only IP addresses, this connection between Web clients and sensor devices can not be maintained.

10.3.1.5 Implementation

To demonstrate the feasibility of this approach, we emulated our own .env DNS server using BIND[9], which is the most widely used DNS software on the Internet, which implements the DNS standard. We also simulated hundreds to thousands of sensor devices, which declare their Web presence to the DNS server. Each device offers some random environmental service and is located at a random location around the world. We recorded the time needed by the .env DNS server to answer PTR requests for random *service.location.env* requests by Web clients, when an increasing number of sensors was registered to the DNS server. The results are presented in Figure 10.2.

As the figure shows, the average response times scale particularly well as the number of registered sensor devices is increased. In the case of 1 million registered devices, average response time is only 16 ms. Network delay is negligible, since the Web clients in this experiment are located in the same local network with the .env DNS machine.

Of course, our experiments with 1 million sensor devices are very small in relation to the billions of physical devices that are expected to flood the Internet in the near future. According to CISCO[10], it is estimated that 50 billion devices will be connected to the Internet by 2020.

[9] https://www.isc.org/software/bind

[10] http://blogs.cisco.com/news/devices-devices-everywhere-infographic/

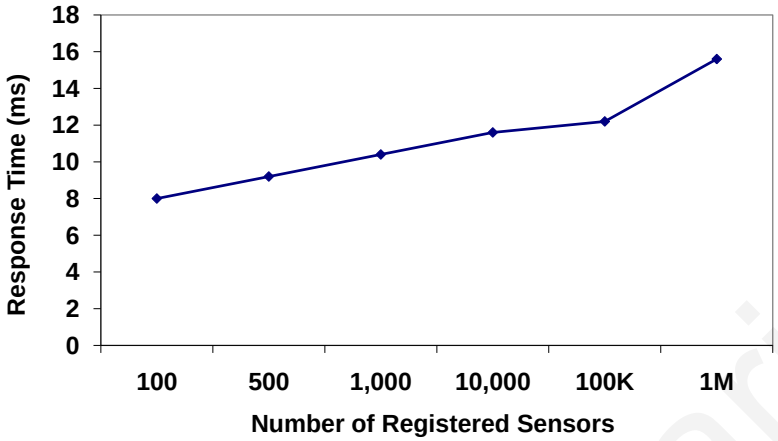

Figure 10.2: Response times of an experimental .env DNS server for PTR requests.

Thus, more extended experimentation is needed in order to consider the scalability of the DNS mechanism and its capability to support tens of billions of embedded devices.

10.3.1.6 Discussion

The procedure of discovering environmental services can be automated by selecting automatically some sensor device from the list returned by the .env DNS server, parsing its WADL file and constructing HTTP requests.

This process can even be personalized, by selecting only devices that meet particular user preferences. For example, some users may wish to interact solely with devices having positive online feedback or those belonging to well-known authorities such as governmental organizations. User characteristics could be extracted from their online social networking status.

Since the whole idea is participatory-based, only people who are willing to share their sensor devices with the online community would do so. Smart homes that are enabled to the Web are expected to have a fundamental role in enabling this concept. In the future, a culture could be created around the concept of sharing environmental services with the rest of the world.

Even though this proposal targets environmental services, it could be well generalized to support any kind of physical devices and/or pervasive services. To achieve this, standardized "domain vocabularies" need to be created, for facilitating the construction of queries by end users. For example, a user that wishes to park his car in Nicosia could just need to type in his Web browser *parking.cars.Nicosia*.

Finally, defining extended environmental ontologies would encourage automatic information retrieval, generalized inferences and advanced Web mashup development (including physical and urban mashups) very easily. For example, when the temperature in Porto is obtained, then the general temperature of Portugal can be automatically inferred.

To sum up, this section investigates in a preliminary level how the DNS could be extended to support global discovery of environmental services. Our small research [96] indicates that it may be feasible to achieve automatic, real-time discovery of sensor devices, by means of specialized domain name servers. However, our thoughts constitute only a novel idea and it would require a great willingness from Web engineers to be realized.

10.4 Advanced Knowledge Inference in Urban Environments

Urban environments are becoming largely crowded, hosting more citizens than they are able to support. Already, urban areas host around 50% of the world's population while the cities' population is expected to double in the next 40 years [163]. The increasing urbanism creates serious ambient problems that downgrade the quality of life of the citizens.

In the future, home residents shall possess in their houses a considerable number of embedded sensor devices, which would measure with high precision the local environmental conditions. The real-time measurements of these sensor devices (except when they measure sensitive data) could be shared by the local (or even the global) community. These measurements could be temperature, humidity, radiation, electromagnetism, pollution, chemicals etc.

As the Web has effectively penetrated in our everyday lives, it can constitute a large-scale and ubiquitous platform, for supporting these massive amounts of sensors that will flood cities in the future. This future abundance of Web-enabled sensors needs to be harnessed toward benefiting people. Web-enablement of smart homes and physical devices could offer an interoperable city-scale ecosystem that involves heterogeneous physical devices and services, shared by home owners. In this context, Web-enabled smart homes would constitute the foundational elements for shaping the next-generation digital cities.

The WoT can be considered as a real-time platform, for supporting citizens to interact with their cities, by means of Web-enabled sensors and actuators. Sensory services can help people to ensure their health and safety, and also to trust their environment. Environmental awareness has the potential to help people become active members of their urban landscape.

10.4.1 Web-based Global Sensor Discovery

City-scale environmental awareness can only function if sensor devices can be automatically discovered along with their exact location. Standardized Web protocols for real-time sensor discovery as well as a search engine for the WoT are currently unavailable. A novel pervasive technique for automatic discovery of Web-enabled environmental services is proposed in Section 10.3, however, it is still in an experimental stage.

As mentioned earlier in Section 10.3, approaches such as Dyser [126] and Snoogle [172] need a long way before they become efficient engines for the real world. Their large acceptance and popularity are hindered by the many security and privacy issues that need to be solved, before convincing people to contribute with their own devices.

Until then, online global sensor directories can be employed for sensor discovery. One of the most well-known directories available today is Cosm [82] (called Pachube until recently). Cosm enables people to share, discover and monitor in real-time environmental data from sensors that are connected to the Web, around the world. The main drawback of Cosm is its centralized nature.

Decentralized approaches such as IrisNet [63] use a hierarchical architecture for a global sensor Web. More advanced, systems such as G-Sense [131] support a peer-to-peer infrastructure for global sensing and monitoring. Although Cosm has the drawback of a single point of failure, it is very popular, being employed by thousands of people and companies worldwide. Besides, it offers a clean Web API that offers advanced services to developers and users. In the following subsection, we exploit the functionality of Cosm to develop a mobile application that exploits sensory information to inform the citizens about the environmental conditions in their urban environment, helping them to achieve advanced knowledge inference.

10.4.2 Case Study: UrbanRadar - Towards Real-Time Digital Cities

The quickly expanding ecosystem of Web-enabled sensor devices could flood the future cities with massive sensory information about every possible aspect of everyday life. Location can be the crucial element that would facilitate this filtering of abundant sensory information, since citizens would be interested only in their nearby environmental conditions, while they are wandering inside their cities. They can utilize their mobile phones to discover their exact position and discover

through the Web nearby sensor devices, which are enabled to the Web by Web-based smart home applications (e.g. the application framework for smart homes).

UrbanRadar[11] [95] is a mobile application we developed that discovers, locates and interacts with services, provided by Cosm-enabled sensors that are deployed in the vicinity of the user. It is freely available for download in Cosm applications repository[12]. The exact user location can be inferred from GPS services, offered by most new mobile phones[13]. Proximity is defined as a circle with the user's location at the center. The radius can range from tens/hundreds meters to hundreds of kilometers. In highly dense areas such as big cities, a radius of tens meters could be enough while in suburbs this radius can cover few kilometers.

Figure 10.3 illustrates the operation of UrbanRadar. The mobile phone represents the user's position and the red circle indicates the area of interest, in which the user can be informed about sensory services. According to the figure, one Web-enabled Telosb sensor mote (see Section 5.1.1) exists in this covered area, with which the user can interact and be informed about its sensing services/measurements.

Moreover, the UrbanRadar application supports the creation and management of urban mashups (see Section 4.2.7). Since every person perceives differently his environment, urban mashups should be adjusted to the particular needs of each mobile user. Consequently, UrbanRadar has been designed to allow users to create easily their own urban mashups.

The procedure of deriving the impact of each sensory service to the overall inference about the occurrence of some mashup, depends on how each citizen understands his environment. Mobile users are encouraged to specify themselves the impact of each physical service to the triggering of

[11] http://apps.pachube.com/pachuradar/

[12] https://cosm.com/apps

[13] In general, other localization techniques can be utilized (e.g. Wi-Fi positioning, localization based on GSM Cell ID). We selected GPS because it provides high accuracy.

Figure 10.3: The operation of *UrbanRadar* mobile application.

some urban mashup by defining weights, expressed using simple keywords such as *high*, *medium*, *low* etc. Mobile users can also adjust the *freshness time* of each environmental measurement. For example, a measurement of temperature could be valid for some hours while a measurement of noise would be valid only for some minutes. Finally, users may define the *sensitivity* of each urban mashup, to determine when they should be notified about a possible mashup triggering.

Overall, this application investigates the possibilities for advanced knowledge retrieval in urban pervasive environments by means of urban mashups. We believe that mobile applications such as UrbanRadar can contribute in the active involvement of citizens with their urban landscape. By interacting with their local environment, people would become more aware not only about their house, but also about their town, ensuring a better quality of life. The concepts of UrbanRadar and urban mashups can constitute drivers towards the vision of digital cities.

10.5 Securing Web-Enabled Smart Homes

Security is a fundamental aspect in smart home solutions and it becomes even more important when considering an open, Web-based environment. We need to note that this section only discusses some general aspects regarding assuring security in Web-based smart homes. These aspects include defining some basic requirements for security, identifying some popular potential attacks and considering security threads both in-home as well as between smart homes and third-party applications, such as social networking sites (see Chapter 8), Web resources declared in physical mashups (see Section 4.2.6) and low-level smart grid controllers for integrating smart homes to the smart grid (see Chapter 9). A full study, which is beyond the scope of the current thesis, is required in order to study in detail all the security issues that result in a Web-based architecture for smart homes.

Security is a sensitive topic that does not tolerate any compromises. For example, since the operation of smart appliances/smart power outlets (see Section 2.1.2) needs to be managed by an energy-aware smart home application (e.g. the application framework for smart homes - see Section 4.2), it needs to be ensured that these smart devices are managed by the appropriate home application and not the application running on a neighboring home or by an attacker. Management of these devices could cover important tasks such as turning on energy-consuming electrical appliances during off-peak periods (see Section 9.2).

Moreover, the bidirectional Web-based communication between smart homes and third parties (e.g. low-level smart grid controllers, social networking sites) requires a trustworthy communication environment, where each party trusts the other communicating party, as well as the correctness, integrity and freshness of the received data. For instance, considering a smart grid scenario, upon reception of pricing messages from the utility, the application framework for smart homes

is assumed to take some actions (e.g. to charge an electric vehicle). If the pricing messages were changed on-route, the application will possibly take wrong actions. This could cause financial losses for the consumers, and even lead to a power outage (e.g. by sending fake low-cost tariffs during peak-periods).

10.5.1 Requirements for Security and Potential Attacks

To provide secure communications in Web-enabled smart homes, the following basic security services need to be guaranteed:

- **Authentication**. Ensures the identity that another party claims to be. For example, the smart home application framework needs to be sure that some home member is the one who attempts to interact with the home environment.
- **Integrity**. Ensures that stored or received data were not modified on-route. For instance, the application framework needs to ensure the integrity of metering data received by smart power outlets.
- **Authorization**. This service allows one party to verify that another authenticated party has the right to do some actions or access some resources. For instance, the framework needs to be sure that a tenant requiring access to some electrical appliance has the necessary rights.
- **Confidentiality**. Ensures that data is illegible to non-authorized parties. For instance, the energy consumption information consumed by electrical appliances and transmitted to the framework by smart power outlets needs to be encrypted, such as only the home residents would be able to access it.
- **Non-repudiation**. This service prevents one party to deny sending a message or doing some action. For example. in a demand response program of a smart grid scenario (see

Section 9.2), assuming that a utility sends a pricing message with a low price Y but applies a high price Z, it could not deny the fact that the low-price message was really sent by it. Furthermore, the utility needs to be sure that a consumer could not contest a bill by denying the sending of the corresponding energy measurement (see Section 9.4).

- **Freshness**. This service protects from replay attacks, where a valid message sent at time t, is also sent in the future by the attacker. For instance, an attacker could replay events that occur inside the smart home, such as the opening of a door or movement inside some room of the house.

- **Pairing**. We need to guarantee that communications involve only authorized smart home devices belonging to a smart home A, and not those of a neighboring smart home B. In addition, smart devices of home A must be managed solely by the smart home application deployed at smart home A. Also, we need to guarantee that the smart home application in home A controls only its own home devices, and not those of a neighboring home B. The pairing will definitely require some level of user interaction, depending on the I/O and computational capabilities of the paired devices.

Unfortunately, the Web/HTTP does not provide a trusted communication environment, since it offers poor built-in security mechanisms and, as a consequence, needs to rely on some extra mechanisms to provide stronger security. For instance, HTTP is a potential target of several attacks that could harm the proposed Web-based architecture, such as:

- **Man-In-The-Middle (MITM) attack**. An attacker successfully impersonates each communicating party (e.g. smart home framework and home devices) to the other party, and injects, modifies or drops packets. This attack could target operations of the house such as home automation, physical mashup execution, eventing, task scheduling etc.

- **Man-In-The-Browser (MITB) attack**. This attach involves a malicious program (e.g. a Trojan) that infects a Web browser and takes control of data entered by the user or data retrieved from the Web server and displayed by the browser. This attack could harm the user by displaying false statistics about his detailed energy consumption, and not the real statistics provided by the actual smart meter/smart power outlets of the house.

- **Denial of Service (DoS) Attack on HTTP**. This kind of attack aims to make a service or resource (e.g. a Web server) unavailable. For instance, this attack could target making sensor and actuator embedded devices unavailable, breaking for example the security alarm system of the house. A similar attack could be launched against the smart home framework, making home automation operations unavailable at the victim smart homes.

10.5.2 Security Inside the Smart Home

Assuming all the in-home interactions are done through the Web, resource-constrained devices such as sensors, actuators, smart power outlets and smart appliances shall run an embedded Web server, in order to expose their capabilities as Web resources, accessible by the application framework for smart homes. The main challenge becomes how to secure this Web-based interaction against attacks, in such a resource-constrained environment.

Despite recent achievements in IP-enabling embedded devices, the WoT is not yet very common in embedded systems and devices. Additionally, using standard security protocols such as Transport Layer Security (TLS) [41] or IPsec [99], does not yet constitute a popular practice in embedded computing, due to the heavy induced costs on the device operation.

Recently, various efforts have been made to make security protocols feasible for resource-constrained devices. Since the majority of smart home devices inside the house typically use IEEE 802.15.4 at the PHY and MAC layers (see Section 2.1.5), they may use the cryptographic

facilities provided by the MAC layer, and in particular the Advanced Encryption Standard (AES) [9] algorithm to provide both encryption and data integrity. AES is a symmetric-key algorithm that supersedes the Data Encryption Standard (DES). It is now used worldwide for encryption of electronic data. For instance, 6LoWPAN [119, 107] employed the AES algorithm for achieving effective in-home security. However, the key provisioning and key management are still open issues, and need to be determined at higher layers.

As another example, CoAP [150], as a lightweight protocol for information transfer in sensor networks that shares several similarities with HTTP, involved CoAP security for securing the home environment. CoAP security [17] is based on the use of Datagram TLS (DTLS) [141] or IPsec for securing the communication between the client (e.g. the smart home framework) and the server (e.g. the home devices) at the transport and network layers.

DTLS is a variant of TLS that operates over UPD (TLS operates over TCP), and which allows the establishment of a secure communication channel over which CoAP messages could transit. Due to the constrained environment of physical devices, only a subset of encryption and hash functions is assumed, in addition to the use of Elliptic Curve Cryptography (ECC) [81] instead of classical public-key cryptography (e.g. RSA, DSA), due to its lower overhead in computation, storage and bandwidth.

To sum up, secure communications inside the smart home environment may be enabled through 6LoWPAN or CoAP and secured by AES or DTLS. The 6LoWPAN/CoAP request/response messages are exchanged in a way similar to HTTP, in addition to an application-level acknowledgment, since TCP is not used at the transport level for reliable data transfer. We strongly recommended that future real-life deployments of energy-aware smart homes need to consider protocols that provide secure communications, such as 6LoWPAN and CoAP.

10.5.3 Security between the Smart Home and Third Parties

Securing the interaction between Web-based smart homes and third parties is important for blending the Web-based home functionality with other Web entities and resources in a secure, feasible and acceptable way. This interaction includes exposing the functionalities/services offered by smart homes (see Section 5.2.3) as Web resources, accessed and utilized by third-party applications. In addition, these remote applications may expose a set of information as resources, accessed by smart homes via the Web. For example, an electric utility could provide its energy tariffs as Web services in near real-time.

By leveraging the existing Web security mechanisms, several security issues inherent to such Web-based interaction need to be addressed. Using HTTP Secure (HTTPS) [140] is one way of protecting the communications between the two parties. HTTPS is HTTP layered over the TLS protocol. The TLS protocol [41] fits between the application and the transport layer (mainly TCP), and provides a number of security services over the Internet, such as cryptographic key-exchange and per-session key establishment, mutual authentication, integrity, confidentiality, non-repudiation and freshness.

TLS allows the establishment of a secure communication channel between a TLS-enabled client and a TLS-enabled server, in which the server is first authenticated through a certificate (optionally also the client), and then secret session keys are established using some key management protocol. Once the communication channel is established, HTTP request/response messages may be securely sent between the smart home application framework and the other Web entity.

Since the application framework for smart homes and the third-party application (e.g. a social networking site, a low-level smart grid controller) play both the role of HTTP client/server, mutual authentication through public-key certificate is mandatory for TLS use. Each smart home may

obtain a pair of certified private/public keys from a trusted Certificate Authority (CA), with a strong advice to keep the private key on a smart card, onto which all public-key cryptographic operations are done, in order to avoid the private key disclosure.

For reinforcing smart home security, the equipment on which the framework is running should not be visible or directly accessible from the outside. Furthermore, smart homes should be protected by a firewall, to prevent possible Denial of Service (DoS) attacks. In this way, the firewall will be the first line of defense for the house, while the access to the resources provided by the application framework could be easily done through redirection rules implemented at the firewall. Moreover, other Web entities that interact with the smart home may also obtain certified private/public keys, and securely disseminate their certificates to the respective smart homes.

Since the smart home application framework and the other Web-based parties are assumed to be hosted in powerful machines, the overheads induced by the TLS protocol (e.g. computation, transmission) could be affordable. However, if the home framework is running on a resource-constrained computing device of the house (e.g. a home router, a smart meter), then the implementation of the whole TLS protocol could not be very efficient, because of increased memory demands and expensive TLS cryptographic operations, especially during the hand-shake phase for key-establishment. Hence, in this case, a trade-off exists between performance and sufficient security when employing HTTPS.

Chapter 11

Conclusion

In this thesis, we investigated the general process needed in order to enable a smart home environment to the Web. This Web-enabling procedure was examined from various aspects, having in mind the relation of the smart home as an entity with important and influential actors of everyday life, now and/or in the near future, such as online social networking, the smart grid of electricity and the urban environment.

An important element in this thesis was the development of a Web-based application framework for smart homes, which can constitute a foundational pillar towards enabling a true Web-enabled home environment. The ”citizens” in the experimental smart home we implemented were sensor motes, measuring the environmental conditions inside the rooms of the house and smart power outlets, used to control electrical appliances and measure their electrical consumption. These home devices provided the means towards automating our proof-of-concept house. Their Web-enablement process, including their discovery, service description and interaction, was a challenging procedure as the goal was to achieve a ”plug and play” ecosystem inside the house.

Overall, our proposal constitutes a flexible application-level solution for home automation, based on combining existing Web technologies and reliable, well-studied Web techniques. Our

work may be considered as a promising practice for home automation, supporting multiple simultaneous home residents. By following the physical mashup paradigm, our approach facilitates this procedure and it is characterized by high interoperability, supporting heterogeneous home devices and services. Through various case studies, we demonstrated that application development becomes flexible and easy by means of the application framework, by harnessing well-known and understood Web principles.

An important enhancement in the functionality of the application framework was the integration of request queues, enabling multiple concurrent users and efficient management of the communication with the physical devices of the smart home. A detailed analysis of the system was performed and potential advantages were identified and discussed. A significant aspect of this mechanism is the ability of fast retransmissions, in case transmission failures occur in the wireless medium of the home environment. This mechanism was used to mask effectively device and transmission failures, increasing the robustness and fault tolerance of the application framework, offering certain reliability.

The use of intermediate request queues, associated with embedded home devices, also facilitated the implementation of a priority-based request scheme, supporting prioritized requests for time-critical tasks. Furthermore, load balancing was employed for better serving high traffic from numerous, simultaneous home residents. Harnessing request queues encourages further system analysis by employing some basic aspects from queuing theory.

Our technical evaluation efforts indicated that, by following the Web model, satisfactory performance of household embedded devices can be achieved, in terms of response times and energy consumption. Our findings showed that IPv6/6LoWPAN can be well applied on sensor motes without prohibitive overhead on the device operation. Especially when combined with

Web caching, IPv6/6LoWPAN can even outperform (identical) applications that utilize the native messaging stack of TinyOS, concerning response times. Event-based Web messaging (push communication), as well as Web caching, may be applied for preserving the battery lifetime of sensor devices, while offering faster notifications in case events are triggered. Event forwarding in the case of a streaming-based wireless sensor network also proved to operate efficiently, in small numbers of physical devices and subscribers.

In general, our evaluation results showed that the application framework offers acceptable scalability (for typical smart home scenarios), supporting large numbers of residents interacting simultaneously with their home environment as well as numerous embedded home devices. Using a typical laptop for hosting the framework, up to a hundred simultaneous tenants may be supported with acceptable performance (assuming that each tenant makes one request per minute), while much larger numbers of home members are not realistic for smart home scenarios. In addition, a few hundreds of home devices may be effectively supported with satisfactory performance.

Furthermore, various interesting Web applications, relevant to smart homes were developed, in order to encounter technological and societal challenges such as energy awareness and conservation. These applications reused the functionality of Web-enabled smart homes, offered as a Web API, with entities and resources available on the Internet/Web. For example, social networking sites were used for sharing home devices and their services between family members, social competitions were performed for increasing the energy awareness of residents that lived in blocks of flats, and Social Electricity was presented, as a novel Facebook application that helps people to understand their electricity footprint by means of social and local comparisons with their friends and their neighborhood.

As another demonstration, the integration of smart homes to the smart grid of electricity through the Web was examined. Smart grid controllers could interact in real-time with Web-based smart homes through their Web API, to achieve synchronized operation of smart homes and the smart grid. In such an integration, residents would save money by exploiting the demand response feature of the smart grid, for scheduling electricity-related tasks for low-tariff periods of the day, while the smart grid could guarantee frequency stability of the system and prevent frequency abnormal operation by employing load shedding.

Then, we explored how the Web-enabled smart home, as an important element in future digital cities and societies, may contribute for addressing various challenges. Some of these challenges include global discovery of environmental services in real-time through the Web and increased environmental awareness and knowledge inference in the urban area. Another aspect beyond the smart home research covers the extension of the application framework for smart homes, as a powerful and flexible middleware for use in numerous pervasive scenarios.

Definitely, a great challenge for future smart homes is also security. Security in smart home environments is a large topic and extended Web-based schemes need to be developed that involve end-to-end security mechanisms that guarantee authentication, data integrity and confidentiality, from the home devices to the application framework, and from the application framework to third-party Web entities. We dedicated a small section discussing security inside the smart home and beyond, touching briefly only some general aspects.

The rest of the chapter is organized as follows: Section 11.1 summarizes the contributions of this thesis and Section 11.2 explains how the requirements for future smart homes, as defined in Section 4.1, have been met through this work. Then, Section 11.3 discusses the overall potential of smart homes developed using Web principles, presenting general perspectives of the general challenges that would appear in smart home research in the near future while Section 11.4 identifies

some relevant open issues which have not been addressed in this dissertation and are important in future initiatives. Finally, Section 11.5 defines future work in Web-enabled smart homes.

11.1 Summary of Thesis Contributions

In summary, the main achievements of these thesis are the following:

- The development of a Web-based application framework for smart homes.
  - High interoperability and support of heterogeneous home devices and services.
  - Support of multiple simultaneous home residents and hundreds of physical devices.
  - Addressing robustness and reliability issues such as device and transmission failures.
  - Promotion of easy and flexbile application development for smart homes by third parties, following the physical mashup paradigm.
  - Efficient and reliable management of the communication with home devices using request queues.
  - Support of prioritized requests and load balancing.
- Addressing issues that concern the Web-enablement of home devices, including their discovery, service description, sharing and interaction.
- Evaluation efforts that indicate satisfactory performance of IPv6-enabled home devices, in terms of response times and energy consumption.
- Improvement of performance of home devices by adopting Web techniques such as Web caching and event-based Web messaging.
- Acceptable scalability for typical smart home scenarios, up to a hundred of home residents an a few hundreds of embedded home devices.

- Integration of smart home applications with online social networking sites, for sharing home devices with the rest of the family, other relatives or close friends.
- Social competitions between neighboring houses targeting energy conservation.
- Social Electricity, as an innovative Facebook application that helps people to understand their electricity footprint by means of social comparisons with their friends and and local comparisons with their neighborhood.
- Demonstration of the integration of Web-based smart homes to the smart grid of electricity through the Web, in order to enable energy-related applications such as demand response and load balancing.
- Addressing (only) some basic, general aspects that concern security in the smart home environment and beyond.
- Finally, investigating how Web-enabled smart homes, as an important element in future digital cities and societies, may contribute for addressing various real-life challenges.

Before proceeding, it is important to note that our proposal has also some limitations. At first, the HTTP protocol implies a request-response interaction model (based on the classical client-server pattern) for the communication with home devices. This is not the most suitable in event- or streaming-based applications. We managed to address this issue by employing event-based Web messaging through a pub/sub approach. Additionally, our analysis and evaluation efforts assumed a static deployment of home devices. Mobility was not considered and it might be a basic characteristic in some other pervasive scenarios. Finally, the application framework may not scale for hundreds or millions of users. Although this was not a requirement in this thesis, it would

be beneficial to assess experimentally the scalability capabilities of the framework, to determine also its applicability in other pervasive application scenarios.

11.2 Meeting the Requirements for Future Smart Homes

In Section 4.1, we listed some requirements which are important in future smart home solutions that aim to address the heterogeneity of home devices. These requirements concern the interaction with the devices, the design and development of the application framework for smart homes, as well as some performance constraints. In this section, we discuss how we met these requirements through the work in this thesis, by listing each requirement with the corresponding actions for addressing it:

- **Multiple interaction possibilities with home devices**. We have developed 3 different interaction patterns with physical devices: ad hoc interaction, through the classical client-server model of the Web (see Section 5.2.3); event-based push messaging, when events are sent only when something important happens (see Section 5.2.4); and continuous monitoring, when devices stream data at regular intervals (see Section 7.4).

- **Multi-hop wireless communication** between home devices, forming a WSN in the home environment. By adopting Telosb sensor motes and Ploggs smart power outlets (see Section 5.1), we have enabled an IPv6-based sensor and actuation network in a proof-of-concept smart home deployment. 6LoWPAN and ZigBee were the underlined communication protocols used for multi-hop wireless communication inside the house.

- **Plug and play** home devices. We proposed and implemented Web-based patterns for automatically discovering physical devices, describing their services, interacting uniformly with them and sharing their functionalities through online social networking (see Section 5.2).

- **Ad hoc interaction** with home devices. The design of the application framework for smart homes facilitates the communication with physical devices/services just by implementing some basic methods for sending/receiving messages to/from the devices (see Section 4.2.1).

- **Uniform access** to heterogeneous devices. This is achieved by transforming home devices as embedded Web servers and by exposing their capabilities as RESTful Web services (see Section 5.2.3).

- **Particularities of resource-constrained environments get abstracted**, offering some reliability. Achieved by handling these failures on the application framework for smart homes (see Section 4.2.4).

- **Masking transmission failures** by means of request queues, associated with home devices on the application framework (see Section 6.4.2).

- **Data from devices is open**, easily exported into third-party applications in standard formats (e.g. XML, JSON). This was enabled through a RESTful modeling of the services offered by home devices (see Section 5.2.3).

- **Interoperable programming interfaces** provide the primitives to end users to perform advanced tasks. Achieved by following the physical mashup example, promoting the creation of smart home applications in any language supporting TCP/IP (see Section 4.2.6).

- **Graphical user interfaces** that facilitate the management and control of the smart home environment have been developed, allowing interaction with home devices, development of smart rules, task scheduling etc. (see Section 4.3).

- **Real-time feedback** of the energy consumption of electrical appliances was achieved through the use of Ploggs and by integrating graphical Web interfaces on top of the application framework (see Sections 5.1.2 and 4.3.3).

- **Layered architecture and flexible design** of the application framework, based on Web technologies and REST principles (see Section 4.2).

- The application framework has a **lightweight implementation** as it needs only 2.7 Mbytes for its full installation (see Section 4.2).

- **Direct access for residents** to their home environment, through the Web (see Section 4.2).

- **Support of multiple residents**, who may interact simultaneously with the smart home through the Web, is achieved by using request queues (see Section 6.4.1).

- **Support for prioritized requests** from home residents has been developed by extending the (FIFO) request queues into priority heaps (see Section 6.4.5).

- **Support for streaming events** to interested Web-based third parties has been integrated on the application framework, allowing multiple events to be sensed and forwarded simultaneously to multiple subscribers (see Section 7.4).

- **Small waiting times for concurrent requests** from various home residents are obtained through load balancing and prioritized request execution (see Sections 6.4.4 and 6.4.5).

- **Reliability** in regard to request satisfaction is obtained by means of employing failure handling mechanisms and request queues (see Sections 4.2.4 and 6.4.2).

- **Acceptable performance in terms of response times** from IPv6-enabled embedded devices, which are in average less than a second even in high traffic from tens of home tenants, has been achieved (see Section 7.2). By employing Web caching (see Section 4.2.5), response times are further decreased.

- **Battery lifetime** of devices is prolonged by Web techniques such as Web caching and event-based Web messaging (see Sections 4.2.5 and 5.2.4).

- **Acceptable performance in terms of push times**, which need to remain less than a second, regardless of the number of subscribers/events per second. This requirement was partly satisfied, obtaining push times of less than a second only when subscribers are less than 50 (see Section 7.4). Nonetheless, a larger number of subscribers is unlikely in typical houses.

- **Extensibility**. The system needs to evolve with minimal coupling between all of its components (devices, users, applications etc.). The evaluation efforts in Chapter 7 indicate that the system evolves with tens of residents interacting with their smart home while 6LoWPAN may handle effectively increasing numbers of home devices. Different routing schemes may increase the extensibility of the IPv6-based WSN even more [106].

- **Scalability in terms of home residents** and requests is partly supported by the application framework (see Chapter 7), which handles effectively the operations needed when tens of home tenants interact with the home environment.However, when hundreds of residents are involved, the framework cannot satisfy their requests in satisfactory time. Thus, in its current form, the application framework is not suitable for smart building scenarios with hundreds of tenants/workers.

- **Scalability in terms of home devices** is supported by the framework, considering a few hundreds of embedded devices (see Section 4.2.2).

11.3 Discussion and Outlook

In this section, we discuss the general future potential of smart homes that operate using Web technologies. As we demonstrated through various case studies, the Web-based operation of smart homes may be seamlessly blended through the Web with third-party entities in order to solve societal, environmental and technological issues. These third-party entities include online networking sites, the smart grid of electricity, online sensor directories, even the DNS system for global discovery of environmental services. This interaction with Web entities can contribute in solving problems that concern energy awareness, energy conservation, pervasive service discovery, sharing of physical devices/services between friends etc.

It is important to note that in this thesis, we do not propose a new technology for advanced home automation, nor an effective, optimized approach for energy conservation. Our focus is to show how easy and elegant home automation can become, when employing Web technologies. We claim that by some of the presented applications (e.g. energy-efficient smart rules, social competitions in blocks of flats, Social Electricity Facebook application, exploiting demand response program of the smart grid), our proposal can contribute in saving energy, but alone it would not be capable of effectively reducing the overall electricity consumption. We recognize that home residents are not expected to have the expertise to design energy-efficient strategies, even though their active involvement in energy-saving initiatives can engage them in sustainable lifestyles [138].

Nonetheless, our application framework for smart homes can be utilized as a flexible platform, on top of which more advanced practices for energy-efficient home automation can be developed. Classical optimization techniques, or even machine learning techniques such as for example neural

networks could be applied, for the creation of advanced energy-saving schemes. Numerous European projects [7, 8] started to employ artificial intelligence and machine learning for advanced home automation. The application framework can sit between the physical home environment and the intelligent algorithms, to facilitate home automation and energy conservation, by offering a robust and reliable physical ecosystem, which may operate with satisfactory performance.

As we mentioned numerous times through this thesis, we envision smart homes as the foundational elements of future smart neighborhoods and cities. Similar to living systems in nature, houses would act collectively in relation to their ecosystem (e.g. neighborhood or even city) they reside in. This will help to address safety, health-related or sustainability issues (e.g. reduction of carbon emissions). As an example, embedded sensor devices, measuring with high accuracy the physical environment, may expose their functionalities as Web services through Web-enabled smart homes, and contribute in city-scale participation-based crowdsourcing programs for urban environmental awareness, creating environmental maps around the urban area.

The *Semantic Web* is an important standard that could be applied in the future for advanced interoperability between machines in smart home environments. The standard promotes common data formats on the Web, including semantic content. Inside the local home environment, even when numerous heterogeneous devices exist, semantic Web techniques are not necessary, although their use would further contribute in M2M interoperability. However, when the smart home collaborates with third-party Web resources for large-scale application scenarios, the Semantic Web would be suitable for avoiding ambiguities concerning pervasive services and sensory data.

Moreover, future smart homes are expected to upgrade their role and influence to the home tenants. The augmented monitoring and sensing capabilities of home environments, combined with enhanced artificial intelligence of smart home applications, would be used to understand the habits, preferences and characteristics of residents. This understanding will then allow the smart

home to adapt its operation to increase the comfort and safety of people, whilst assisting them in saving energy. More advanced, this understanding of residents' behavior could be blended seamlessly with their everyday lives, through the technical equipment they use through the day. The most common examples are their mobile phones, their cars and their computers at work.

For instance, the air conditioner in the family room may be sharing the temperature preference information of some tenant with his car, for automatically adjusting the temperature inside the vehicle while driving. The smart phone could follow the waking hours of the user to automatically adjust its own alarm, to ensure that the user will wake up early enough to go to work. Or perhaps the MP3 player will update its playlist according to the music preferences of the resident, as they are derived by monitoring the home's Hi-Fi system. In all these examples, the smart home will be at the center of this connected world. A Web-based operation of the smart home would promote its interoperability with the tenant's everyday technical equipment.

According to a highly influential position paper about the changing role of pervasive middleware [25], the trend in middleware is from discovery and orchestration to recommendation and planning. This may be applied also for middleware targeting smart home environments. The research challenges that need to be faced according to the paper are: a common model for representing generalized services and data; algorithms for distributed recommendation, planning and orchestration; reasoning and learning from context; privacy-aware strategies for resource management; and incentive programs for boosting participation.

Overall, we believe that the future in home automation is towards the Internet/Web. Web technologies have the potential to become the future standards in home environments, towards an interoperable ecosystem. Web-enabled smart homes will play a crucial role in the technological challenges of the smart neighborhoods and cities of the future, and they could contribute significantly to develop a sustainable, secure and reliable world.

11.4 Open Issues

Although this work tried to study the Web-enablement of smart homes in a comprehensive way, it was inevitable to leave some issues open for future research. Besides, the research area around the WoT and home automation is very large, and numerous aspects remain to be addressed.

At first, security is a topic that needs more detailed analysis, considering the Web functionality of smart homes. Security needs to be considered both in-home, in regard to the communication with Web-enabled embedded devices but also between the smart home and third parties, such as smart grid controllers and home residents who interact remotely with their house through the Web. We discussed some basic aspects of security in Section 10.5.

Another important issue concerns privacy of the owners and users of the physical devices and services of the smart home environment. Personalized schemes need to be developed that respect the privacy of home tenants, giving them full control over the level of sharing personal data with third parties, e.g. to share the consumption of their electrical appliances with their electric utility. Concerning the Social Electricity project, we performed in Section 8.3.4.2 a small analysis of the different privacy levels of people and their willingness to share their personal electrical consumption data with different groups of people (e.g. family members, relatives, close friends). We suggest the use of privacy circles, where tenants would be able to categorize their contacts in them and associate a different privacy level to each circle. This is of course only a small measure to increase the privacy of home owners. More effective measures are still needed.

An intelligent plan for energy conservation still needs to be developed inside a Web-enabled smart home. While our proposal addresses heterogeneity of home devices, allowing reliable interaction with them and easy development of smart home applications, optimization techniques

and approaches from artificial intelligence need to be applied for the creation of advanced energy-saving schemes that respect the comfort levels of the residents.

Another important aspect is the capability of devices to collaborate towards a global goal without human intervention and control. This is one of the visions of M2M communications, where home devices could exchange information automatically to achieve a common task, e.g. to reduce the overall waste of energy. In such an advanced home ecosystem, even the existence of an application framework for smart homes could be redundant.

Moreover, the semantics of the capabilities and services of physical devices need to be described in a formal, uniform way, avoiding ambiguities. The Web-enabling process of devices facilitates this task, since the services offered by the devices may be well described in standardized formats such as WADL (see Section 5.2.2). However, the capabilities and characteristics of each device still need to be described (e.g. indoor/outdoor, transmission range, battery lifetime) as well as the measurements of their sensing modules (e.g. min/max values, scale, accuracy). Example description languages for sensor measurements are EEML and SensorML. Concepts of the Semantic Web[1] can be employed for defining the semantics of pervasive home devices/services.

Something needed to effectively increase the popularity of smart home solutions is a killer application, similar to the way online social networking increased the popularity of the Web. Many believe that demand response programs from the smart grid would be the killer application we seek. Nonetheless, it is still an open issue remaining to be addressed.

Finally, an issue that needs perhaps to be investigated is fairness. It must be ensured that home automation would not favor those who have the financial capabilities to employ these solutions over those who do not. Home technology should not create unfair advantages to high-income people, but better focus on increasing the comfort levels of residents, offering advanced automation

[1] http://www.w3.org/standards/semanticweb/

and more sustainable ways of life. As an example, we shall avoid a scenario in which high-tech domestic premises are promoted by electric utilities as gold customers, participating in sophisticated smart grid programs, while houses without these capabilities are ignored.

11.5 Future Extensions and Challenges

In this section, we define our future work efforts, based on the current work presented in this thesis as well as the future challenges and requirements in the area of smart homes. Our main future research directions are listed below.

Further Request Queue Analysis.

Request queues may be defined as a dynamic, adaptive system that handles failures in the home environment providing certain reliability. The use of request queues enables the detailed analysis of the system by employing queuing theory. Queuing theory could answer more complicated questions such as the mean number of home tenants in the system or the mean waiting times. It could also estimate the probability of the request queue to be in a certain state. More advanced, queuing theory could define the distribution of the length of the busy periods, the distribution of the sojourn and waiting times, the distribution of the number of arrivals during the service time etc. This theoretic analysis is also a matter of future work.

Future work could also investigate task scheduling, which involves completing executing tasks within predetermined time deadlines and constraints. For example, task scheduling could include an efficient execution of the schedulable electrical appliances of the smart home (see Section 9.1), as soon as a tariff from the electric utility becomes lower than its normal price in a demand response program from the smart grid (see Section 9.2). Request queues could handle reliably these tasks by assigning high priorities to the electrical appliances that need to operate first.

Finally, in this thesis, a device-centric approach was followed for request queues, i.e. every home device was associated with a single queue. In future, a service-centric technique would also be examined, in which a request queue would be implemented for identical services offered by various physical devices. Services could be further categorized in room-level or even floor-level in some smart home or building application. Through this approach, we expect that request satisfaction will become more effective, since the most reliable devices that provide the lowest response times would be selected. Moreover, caching mechanisms would operate better since service-specific cache hits are expected to be increased (see Sections 4.2.5 and 7.2.2).

Support for Mobile Devices and Dynamic Environments.

In this thesis, we only considered a static deployment of sensors and actuators inside a home environment. While this is the typical case for smart homes, it constitutes a limitation of our work, as it hinders the transformation of the application framework for smart homes into a flexible tool, adaptable for various pervasive application scenarios. Hence, in the future we will examine and evaluate the behavior of the application framework when high mobility of home devices is present, in a highly dynamic and ad hoc home environment. We note that mobility is now supported by 6LoWPAN [151], even though it is still an open issue under research.

We will definitely need to reconsider the request queue mechanism, and especially the request queue retransmission interval (see Section 6.3.2). We note the similarity between the request queue retransmission interval (see Equation 5) and the retransmission timer used at the TCP protocol (see Section 6.3.3), as shown in the following equation:

$$\alpha = RTT_{est} + k \cdot RTT_{dev} \tag{11}$$

RTT_{est} is a smoothed RTT time, influenced by past RTT samples, RTT_{dev} represents the RTT variation and k is set to 4. The TCP retransmission mechanism takes into account the increased unpredictability and variable response times that are observed on the Internet. Thus, an adaptive version of the setting of the request queue retransmission interval would be required. This fact encourages future research that concerns further applying (and comparing) these well-studied Internet principles in home environments for added performance.

Exploring the Scalability Capabilities of the Application Framework.

As we noted earlier, one of the limitations of our framework is the lack of scalability to hundreds/millions of simultaneous home residents. This lack of scalability is partly due to the fact that we installed the application framework during our evaluation efforts on a laptop computer and not on some more powerful computing device (see Section 7.1). This relation between the framework's performance and the computing device type on which it runs is observed in the experiment performed in Section 4.2.2, in which different discovery and response times are measured when different computing devices are considered, namely the laptop computer and a small server.

Nevertheless, the current experimental setup indicates that the system does not scale well to large numbers of home residents. Actually, this is not a problem for typical smart home scenarios, but it prevents applying the application framework in other ubiquitous real-life applications. Future work would examine the scalability capabilities of the application framework, in terms of hundreds of concurrent home tenants. This would include installation of the application framework in a more capable server machine, and possible code optimizations.

Finally, scalability in terms of the number of physical devices that may be supported by the application framework is another aspect that could be investigated in the future. The experiment in Section 4.2.2 indicates the capability of the framework to support a few hundreds of home

devices effectively. Moreover, the 6LoWPAN protocol claims that it can support hundreds or even millions of devices in the same subnet. Nonetheless, this scalability feature needs to be proved experimentally, in relation to the operation of the application framework in presence of high workload from Web clients, and also in presence of mobility.

Social Competitions in Neighborhoods.

The social competition for energy conservation in blocks of flats, presented in Section 8.2, had positive initial findings that justify the development of further relevant case studies.

At first, future work needs to include larger case studies, involving also blocks of buildings in more rural areas. Such studies need to be performed to validate the initial findings and consider more confidently the effect of social norms on energy conservation. A longer period of observation may also be interesting, in order to foster the learning effect of the participants.

Long-term influence on residents' behavior is an important dimension not yet explored. For example, considering energy awareness through real-time feedback, the saving effects are persistent mostly when the feedback systems are present [165]. Thus, the influence of the social competition at the blocks needs to be considered also in the coming months.

In this study, we tried to differentiate our social-based efforts for saving energy from real-time feedback techniques. In such a way, we evaluated our approach without direct influence from other factors. Future work could combine continuous energy feedback with a social competition, to examine whether further savings could be achieved. Also, it would be interesting to consider if any other incentives, other than the "ranking and awards" approach, may be most efficient in such social competitions.

Enhancing the Features of Social Electricity.

To make Social Electricity (see Section 8.3) more interesting and useful to Cypriot citizens, we plan to create the *Social Comparisons* program. This program would run on a volunteer basis, counting on the contribution of Cypriot residents. It will be located in another menu, integrated in Social Electricity main interface, encouraging people to add details about their characteristics and preferences, taking into account the most important factors that affect domestic electricity consumption. These factors could include the size of their home in square meters, the total residents of the house, the number of rooms, their heating/cooling equipment etc. In Appendix J we list the most important factors we identified, after a small research, which affect significantly the domestic energy consumption.

By assessing analytically the significance of each of these factors, we would then permit more effective comparisons of energy consumption between people that share similar preferences. We are in the process of collecting information from Cypriot residents about the characteristics of their houses and we plan to correlate these characteristics with their electricity footprint in different time periods of the year. We would definitely respect the privacy of citizens and we will use this information anonymously, only for research purposes.

In general, we believe that Social Electricity constitutes a novel initiative worldwide and we expect that its deployment around Cyprus would help us include a (more) complete analysis in our study, including hundreds of citizens as subjects (see Section 8.3.3). We aim to transform this application into a powerful, robust platform for future case studies related to energy awareness, with an active community that is willing to participate in these future studies.

Online Social Networking of the Real World

Combining our social case studies together would be an interesting task. Such combinations could provide advanced benefits to the people involved in such studies.

For example, Social Electricity (see Section 8.3.1) could be combined with the idea of social competitions in neighborhoods (see Section 8.2). In this case, Social Electricity could be extended into a platform for energy competitions among local neighborhoods. Such an application would be more effective when electric utilities provide (near) real-time electricity measurements from residential smart meters. Moreover, Social Electricity could be further blended with Social Farm application (see Section 10.2), to achieve online sharing of the energy consumptions of farms of similar type (dairy, greenhouse, etc.). In this way, such comparisons could be helpful in the collaboration of the local industry to reduce overall power usage.

Finally, we plan also to extend our social applications into more general and valuable tools for people, in order to support other real-life scenarios such as health monitoring.

A Graphical Content Management System for Smart Homes.

A radical-new idea concerning home automation would be the development of a graphical content management system (CMS) for smart homes. Similar to well-known CMS for Web sites (e.g. Joomla[2] and Wordpress[3]), a CMS for Web-enabled smart homes could revolutionize home automation practice, allowing end users with almost zero programming experience to configure their house according to their needs.

This CMS for smart homes could be installed on top of the application framework, in some computing device of the house (e.g. laptop computer, tablet, router, smart meter) and would

[2]http://www.joomla.org/

[3]http://wordpress.org/

operate as a Web application. Of course, this vision needs great effort in order to be implemented, and it requires the large contribution of programmers worldwide. However, this is exactly how the most well-known CMS for Web sites worked, i.e. they developed their initial platform and then encouraged users to build their own plug-ins, widgets and extensions.

Figure 11.1 visualizes an example graphical CMS for smart homes. Home tenants would be able to upload photos of the rooms of their house and associate their electrical appliances with embedded devices such as smart power outlets and RFID tags. They would also be able to associate rooms of the house with sensor devices that measure the environmental context of the rooms (e.g. noise, occupancy, temperature, illumination). Furthermore, residents could include widgets from any Web resource and correlate them with specific spaces of their home environment. For example, online weather forecast services could be employed (e.g. World Weather Online[4], Yahoo Weather[5]) or online social networking sites (e.g. Facebook, Twitter, LinkedIn), or even electric utilities providing their current energy tariffs as widgets.

Futuristic examples cover the use of some board inside the house or even the fridge as a surface for posting news from SNS or the user using the television in the living room for viewing streaming content from YouTube. These examples could be enabled inside the CMS Web application on top of the photos of the rooms of the house, as they would be uploaded by home tenants.

The Social Water Application.

Our comparisons regarding energy consumption can be well extended to include other resources, such as gas and water. We focus mainly on water, as it is a valuable resource in Cyprus. Since our country has yearly problems with drought, effective water conservation is particularly

[4]http://www.worldweatheronline.com/

[5]http://weather.yahoo.com/

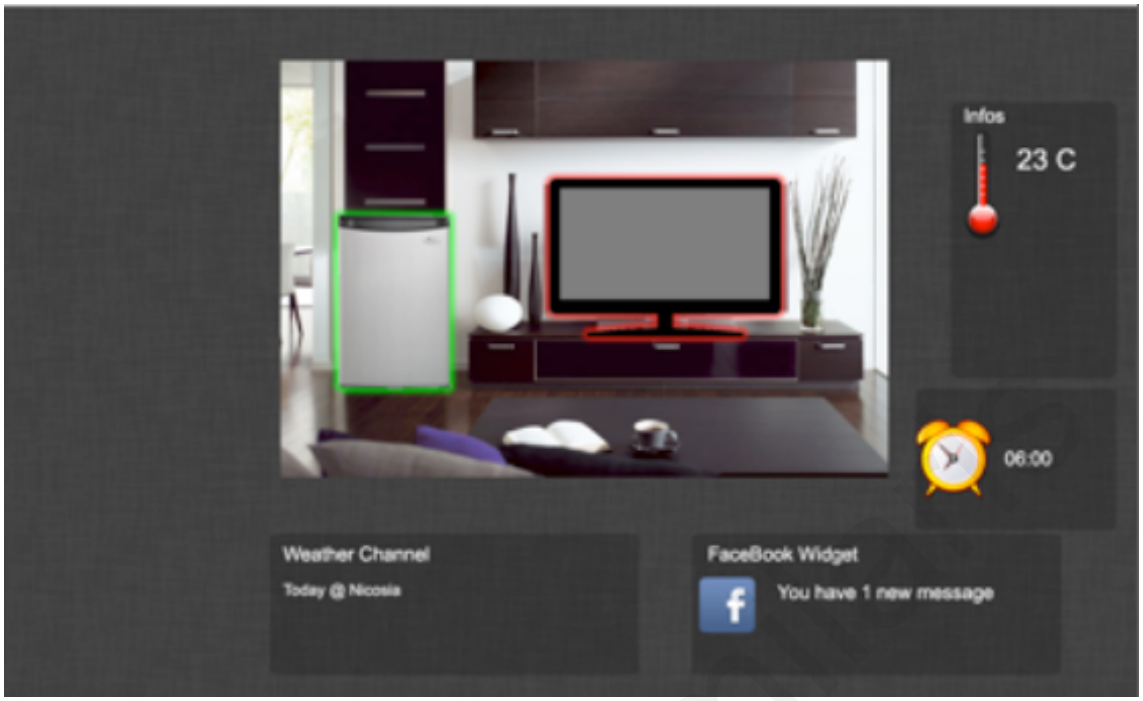

Figure 11.1: A snapshot of an example graphical CMS for smart homes.

important. Hence, we intend to develop *Social Water* Facebook application, following the general concept of Social Electricity application.

Social Water could help people understand their consumption of water through comparisons between their friends, their neighborhood, their village or city. We already started socializing our intentions and had very positive initial feedback. Specifically, the research center Nireas[6], which performs research concerning water quality and conservation, appeared very interested in our efforts and promised to bring us in touch with the main Waterboards of Cyprus[7].

[6] http://nireas-iwrc.org/

[7] The Waterboards are the responsible governmental organizations for the management and distribution of water to the residences. They can provide us with details regarding residential water usage.

Enhancing UrbanRadar Mobile Application.

Right now, we are in the process of enhancing the features of UrbanRadar mobile application (see Section 10.4.2), to apply more advanced automated reasoning for deciding event triggering and urban mashup execution. In the context of this application, simple rule-based automated reasoning such as fuzzy logic [104] can be directly applied. Moreover, it is expected that probabilistic reasoning could provide in general more accurate results and may perhaps prove effective for our model. In this case, the system can be represented as a probabilistic graphical model [22], through which the formulated graph in Figure 4.10 can be used as a belief network [121]. Both fuzzy logic and belief networks will be integrated into the mobile application and their reasoning performance will be validated and compared through our future case studies.

To assess the potential of this application (and of the concept behind its development), we plan to ”simulate” a digital city in a controllable environment. We selected the campus of the University of Cyprus as our targeted urban environment and we are in the process of transforming it into a high-tech space, by creating a number of useful environmental services, targeting students who spend a considerable amount of their everyday lives there. Students will be encouraged to build their own urban mashups, according to their needs and their perception of the environment. After a three-months period, we will study the effectiveness of the urban mashup paradigm through questionnaires and by observing the popularity of UrbanRadar among the people involved. During the evaluation procedure, we will investigate the efficiency of the different parameters that could affect urban mashups, to increase the overall accuracy of reasoning.

Appendix A

Publications

We provide here a complete list of publications stemming from the work in this thesis. These publications are categorized in journal papers, conference papers, book chapters, workshop papers and technical reports.

Journals

1. Andreas Kamilaris, Andreas Pitsillides and Michalis Yiallouros. **Building Energy-aware Smart Homes using Web Technologies**. Journal of Ambient Intelligence and Smart Environments (JAISE), 2012. In Print.

2. Andreas Kamilaris, George Taliadoros, Diomidis Papadiomidous and Andreas Pitsillides. **The Practice of Online Social Networking of the Physical World**. International Journal of Space-Based and Situated Computing (IJSSC), ISSN 2044-4907, Vol.2, No.4, November 2012, (DOI: 10.1504/IJSSC.2012.050007), pp. 240-252.

3. Andreas Kamilaris, Yiannis Tofis, Chakib Bekara, Andreas Pitsillides and Elias Kyriakides. **Integrating Web-Enabled Energy-Aware Smart Homes to the Smart Grid**. International Journal on Advances in Intelligent Systems, ISSN 1942-2679, Vol.5, No.1&2, July 2012, pp. 15-31.

4. Andreas Kamilaris, Vlad Trifa, and Andreas 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, ISSN:1743-8225, Vol.7, No.3, April 2011, (DOI: 10.1504/IJAHUC.2011.040115), pp. 145-154.

Conferences

1. Andreas Kamilaris, Giannis Kitromilides and Andreas Pitsillides. **Energy Conservation through Social Competitions in Blocks of Flats**. In Proc. of the 1st International Conference on Smart Grids and Green IT Systems (SMARTGREENS), Porto, Portugal, April 2012, ISBN: 978-989-8565-09-9, (DOI: 10.5220/0003950301670174), pp. 167-174.

2. Andreas Kamilaris and Andreas Pitsillides. **Using DNS for Global Discovery of Environmental Services**. In Proc. of the 8th International Conference on Web Information Systems and Technologies (WEBIST), Porto, Portugal, April 2012. Full paper published (poster presentation), ISBN: 978-989-8565-08-2, pp. 280-284.

3. Andreas Kamilaris, Diomidis Papadiomidous and Andreas Pitsillides. **Lessons Learned from Online Social Networking of Physical Things**. In Proc. of the Sixth International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA), Barcelona, Spain, October 2011, ISBN: 978-0-7695-4532-5, (DOI: 10.1109/BWCCA.2011.24), pp. 128-135.

4. Andreas Kamilaris and Andreas Pitsillides. **Exploiting Demand Response in Web-based Energy-aware Smart Homes**. In Proc. of the 1st International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies (Energy 2011), Venice, Italy, May 2011. **(Best Paper Award)**.

5. Andreas Kamilaris, Vlad Trifa, and Andreas Pitsillides. **HomeWeb: An Application Framework for Web-based Smart Homes**. In Proc. of the 18th International Conference on Telecommunications (ICT 2011), Ayia Napa, Cyprus, May 2011, ISBN: 978-1-4577-0025-5, (DOI: 10.1109/CTS.2011.5898905), pp. 134-139.

6. Andreas Kamilaris and Andreas Pitsillides, **Social Networking of the Smart Home**. In Proc. of the 21st Annual IEEE International Symposium on Personal,Indoor and Mobile Radio Communications (PIMRC 2010), Istanbul, Turkey, September 2010, ISBN: 978-1-4244-8017-3, (DOI: 10.1109/PIMRC.2010.5671783), pp. 2632-2637.

7. Vlad Trifa, Dominique Guinard, Vlatko Davidovski, Andreas Kamilaris and Ivan Delchev. **Web Messaging for Open and Scalable Distributed Sensing Applications**. In Proc. of the International Conference on Web Engineering (ICWE 2010), Vienna, Austria, July 2010, ISBN:3-642-13910-8 978-3-642-13910-9, pp. 129143.

Book Chapters

1. Andreas Kamilaris and Andreas Pitsillides. **Blending Online Social Networking with the Physical World**. Dynamic Analysis for Social Network (Edited by: C. A. Pinheiro & M. Helfert), iConcept Press, 2012, ISBN: 978-14610987-3-7 (16 pages).

Workshops

1. Andreas Kamilaris and Andreas Pitsillides. **Using Request Queues for Enhancing the Performance of Operations in Smart Buildings**. In the 7th ACM International Workshop on Performance Monitoring, Measurement and Evaluation of Heterogeneous Wireless and Wired Networks (PM2HW2N), in Proc. of MSWiM 2012, Paphos, Cyprus, October 2012, ISBN: 978-1-4503-1626-2, (DOI: 10.1145/2387191.2387193), pp. 1-6.

2. Andreas Kamilaris and Andreas Pitsillides. **The Evolution of Smart Homes-Towards the Web**. In the 2nd Workshop of KIOS Research Center for Intelligent Systems and Networks (poster presentation), Nicosia, Cyprus, April 2012.

3. Andreas Kamilaris, Nicolas Iannarilli, Vlad Trifa, and Andreas Pitsillides. **Bridging the Mobile Web and the Web of Things in Urban Environments**. In the Urban Internet of Things Workshop, in Proc. of IoT 2010, Tokyo, Japan, November 2010 (6 pages).

4. Andreas Kamilaris, Vlad Trifa and Dominique Guinard. **Building Web-based Infrastructures for Smart Meters**. In the Energy Awareness and Conservation through Pervasive Applications Workshop, in Proc. of Pervasive 2010, Helsinki, Finland, May 2010 (6 pages).

Technical Reports

1. Andreas Kamilaris and Andreas Pitsillides. **A Restful Architecture for Web-based Smart Homes using Request Queues**. Technical Report No. TR-12-5, Department of Computer Science, University of Cyprus, June 2012 (28 pages).

Others

1. Andreas Kamilaris and Andreas Pitsillides. **The Evolution of Smart Homes: Towards the Web**. At a two-day workshop organized by the Faculty of Pure and Applied Sciences, University of Cyprus, Nicosia, Cyprus, University Of Cyprus (new campus), November 2012. (poster presentation)

2. Andreas Kamilaris and Andreas Pitsillides. **Building Energy-aware Smart Homes using Web Technologies**. In the 4th Cyprus Workshop on Signal Processing and Informatics, Nicosia, Cyprus, University Of Cyprus (new campus), July 2011. (abstract submission and proceedings)

Appendix B

Application Framework Implementation Details

The application framework for smart homes (see Chapter 4) has been developed in Java using the Eclipse[1] development platform and it follows a modular architecture with three main layers:

1. **Device layer**, which administers embedded home devices.

2. **Control layer**, which initializes, controls and checks the framework.

3. **Presentation layer**, which maintains a Web server following the REST principles.

These three layers are implemented as Java packages and hold the Java classes of the application framework. In other words, each Java class of the framework is categorized, according to its purpose, in one of these three main packages. Below we list the (most important) classes of the application, organized based on the main layer to which they belong.

[1] http://www.eclipse.org/

Device Layer Classes:

- **Devices**. It represents all the devices that have been discovered in the home environment.
- **Device**. It is the thread responsible for managing the communication with some home device.
- **MessageQueue**. It is responsible for implementing the request queue mechanism of each device thread.
- **Request**. It defines the formats and types of the request messages exchanged between the framework and the embedded devices.
- **Response**. It holds the response messages received from physical devices.
- **Resources**. Holds all the services offered by some device.
- **Resource**. A particular service offered by some device.
- **Driver**. Abstract class that defines a standardized interface for physical communication with home devices.
- **TinyosIPv6Driver**. Driver class for enabling communication with IPv6-enabled sensor motes, operating with TinyOS.
- **TinyOSParser**. Parses the request/response messages to/from TinyOS/IPv6-enabled sensor motes.
- **PloggMonitor**. Responsible for the communication and management of Ploggs smart power outlets.

Control Layer Classes:

- **Core**. Acts the main dispatcher of the system, holds the main class for running the framework. Responsible for all the initializations and for maintaining the application's routines.
- **Server**. Maintains HTTP libraries, needed by the Web server.
- **Constants**. This class keeps all the parameters and configurations of the application framework.
- **AsyncToSync**. Helper class for facilitating the transition between synchronous Web operation and asynchronous smart home functionality.
- **WADLparser**. Helper class used to parse the WADL files (see Section 5.2.2), which are transmitted from the home devices during their discovery (see Section 5.2.1).

Presentation Layer Classes:

- **Server**. The Web server of the application framework.
- **RESTApplication**. Routes HTTP requests to the most appropriate Java classes to serve them. It provides REST functionality to the framework.
- **AbstractResource**. Abstract class that generates representations of resources in the standard formats available (HTML, JSON, XML).
- **DevicesResource**. Creates a representation of the physical devices that are available in the home environment.
- **InvokeDeviceResource**. Invokes a request by forwarding it to the appropriate physical device. It sends the response then back to the Web client who created the request.

These are the main classes that compose the application framework. Various smaller classes exist, which only perform some minor tasks. Moreover, the application framework involves a *Simulation* package, which includes classes related to the emulator that is used in the analysis and evaluation procedure of this thesis (see Chapters 6 and 7). An abstract class diagram, displaying the main classes of the framework as well as the relationships between them, is illustrated in Figure B.1.

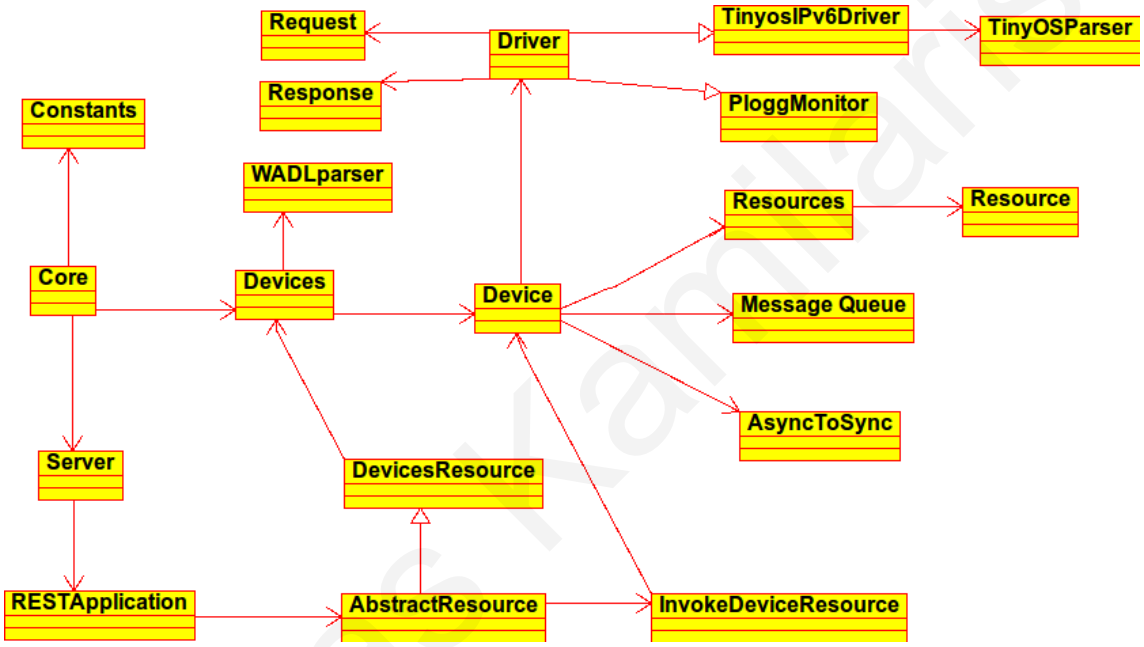

Figure B.1: A class diagram of the application framework.

Appendix C

Application Framework Configuration Parameters

Here we list the configuration parameters that define the operation of the application framework for smart homes (see Chapter 4). These parameters are adjusted inside the *Constants* class, which is located at the package *controlLayer.libraryCode*.

- **REQUEST_RETRANSMISSION_INTERVAL**. Specifies the amount of time of the request queue retransmission interval α (see Section 6.2) in milliseconds.
- **TRANSMISSION_FAILURE**. Defines the percentage of transmission failures during the operation of the framework. It is used mostly for analysis/evaluation purposes.
- **DEV_ALIVENESS_CHECK_TIME**. The interval (in minutes) between the aliveness checks performed by the framework to determine if embedded devices are still alive and operate correctly (see Section 4.2.4). Default value is 5 minutes.
- **DEV_MAX_CACHE_DELAY_TIME**. Defines cache freshness period (in seconds), i.e. the amount of time a cached value remains fresh (see Section 4.2.5). The caching mechanism suspends by setting this value equal to 0. Default value is 10 seconds.

- **DEV_REQUEST_MAX_ATTEMPTS**. Maximum number of retransmission attempts, in case of transmission failures, before a home device is considered unavailable (see Section 6.2). A value of 5 is usually suitable for indoor environments and static topologies.

- **FAILURE_MASKING_MECHANISM**. A boolean parameter that decides whether to employ the failure masking mechanism in case of device failures (see Section 4.2.4).

- **PRIORITIES**. A boolean parameter that specifies whether the application framework employs prioritized requests (see Section 6.4.5).

- **HIGH_PRIORITY**. The integer value assigned at a high-priority request, according to the Algorithm 2. Similar parameters exist for normal- and low-priority requests.

- **PROBABILITY_HIGH_PRIORITY**. Percentage of incoming requests having high priority. Used mostly for analysis and evaluation purposes. Similar parameter exists for low-priority requests. Obviously, the rest requests are labeled with normal priority.

- **PLOGGS_STREAMING_DELAY**. Defines the time interval between electricity measurements of the Ploggs smart power outlets (see Section 5.1.2). Default value is 1 minute.

- **PLOGGS_DISCOVERY_INTERVAL**. Specifies the time interval (in seconds) for scanning the house for new Ploggs smart power outlets. Default value is 5 minutes.

Appendix D

Application Framework Installation Instructions

The application framework for smart homes (see Chapter 4) has been developed in Java using the Eclipse[1] development platform. Hence, it is available as an Eclipse project and may be imported very easily to the Eclipse SDK (File->Import->Existing Projects into Workspace). The source code of the application framework has been released on the NetRL website[2] and on GitHub[3]. The framework is very lightweight, requiring only 2.7 Mbytes for its full installation. The development of the framework and all the analysis and evaluation efforts have been performed using Eclipse Galileo (Eclipse SDK version 3.5.2).

To run the Eclipse project of the application framework, a Java Running Environment (JRE) must be installed on the computing device that hosts the application framework. Furthermore, the Java Communications API[4] needs to be installed to support serial-to-USB communication with any embedded device that would act as the base station, enabling the communication with remote wireless devices. A good tutorial for installing the Java Comm API is available at:

[1] http://www.eclipse.org/

[2] http://www.netrl.cs.ucy.ac.cy/index.php?option=com_content&task=view&id=76&Itemid=54

[3] https://github.com/andreaskami/homeweb

[4] http://www.oracle.com/technetwork/java/index-jsp-141752.html

http://www.agaveblue.org/howtos/Comm_How-To.shtml

The Eclipse project comes along various JAR files, which are needed for its proper operation. These JAR files are:

- tinyos.jar - Used for support of TinyOS 2.x messaging.
- RXTXcomm.jar - Needed for the communication with the sensor mote acting as the base station, from the serial-to-USB port.
- comm.jar - Also needed for the serial-to-USB communication.
- org.simpleframework.jar - Supports HTTP communication with Web entities and XML parsing.
- org.restlet.jar - Supports the Restlet[5] Java-based REST framework.
- org.json-2.0.jar - For JSON encoding/decoding and parsing.
- com.noelios.restlet.jar - Supports a Web server implementation.

Furthermore, the project includes a *wadl* folder, which holds the WADL description files used to describe the services offered by Telosb motes and Ploggs smart power outlets (see Section 5.1).

After the Eclipse project of the application framework is properly imported in Eclipse, the user can run the project as a Java application. The main class for running the Java application is the *Core* class, located in *Control Layer* package. Before running the application, the user needs to set some basic parameters, by navigating to *Run->Run Configurations* and selecting the project.

[5] http://www.restlet.org/

Then, when opening the *Arguments* tab, he must type the following information:

```
<Application Framework Name><Location>
<Domain Name or IP address><Port>{<Device Type><USB Port>}
```

Device Type specifies an embedded technology to be included in the current run of the framework and *USB Port* defines the port of the device that acts as the base station. More than one embedded technologies may be supported. An example setting of the parameters for running the application framework, using both Telosb sensor motes and Ploggs outlets, is the following:

```
ApplicationFramewok UniversityOfCyprus localhost 8080
Telosb USB0 Ploggs USB1
```

Now you should be able to run successfully the application framework and automate your house with reliability and satisfactory performance, with support for all your house members!

Appendix E

Supporting Heterogeneous Device Types at the Application Framework

Example libraries needed for communicating at the application framework with Telosb sensor motes and Ploggs smart power outlets are provided in this appendix section. The case for Telosb motes is shown in Listing E.1 and the Ploggs case in Listing E.2. We note that this source code is not complete, containing the most important functionality for exchanging messages between the framework and the physical devices.

These libraries are basically Java classes that implement the *Driver* abstract interface, which is located inside the device layer of the framework. These libraries are then used by the driver module, to enable physical communication with embedded home devices. The detailed operation of the driver module is discussed in Section 4.2.1.

Listing E.1: Example libraries for communicating with Telosb sensor motes.

```
1 public class TinyOSDriver extends Driver implements MessageListener{
2
3     private MoteIF   mote;    // connects to base station sensor
```

```
4     private short    gateway_tinyOS_id; // app. framework id
5
6     public void startDevice() {
7         mote = new MoteIF(PrintStreamMessenger.err);
8       mote.registerListener(new SmartDeviceMsg(), this);
9         gateway_tinyOS_id = TINYOS_GATEWAY_ID; // (set to 1)
10    }
11
12    public void messageReceived(int dest_addr, Message msg){
13      SmartDeviceMsg smsg = (SmartDeviceMsg)msg;
14      int nodeid    = (int) smsg.get_nodeid();
15      char subject  = (char)smsg.get_subject();
16      short[] sdata = smsg.get_data();
17
18      switch(subject){
19        case 'D':{  // device description message
20          char[] cdata = new char[sdata.length];
21          for(int i=0; i< sdata.length; i++)
22            cdata[i] = (char)sdata[i];
23          String data = new String(cdata);
24          TinyOSParser parser = new TinyOSParser();
25          parser.parseDeviceData(nodeid), data);
26
27          if(devices.containsDevice(nodeid) == false){
28            devices.addDevice(nodeid, new Device(nodeid,name,
29                                    description, location, keywords,
                                        wadl_url));
```

```
30            devices.startDevice(nodeid);      }
31          this.sendMessage(nodeid, 'A', new String());   }
32        case 'R':{ // Service Response message
33          TinyOSParser parser = new TinyOSParser();
34          Response r  = parser.parseResponseData(sdata);
35          int result  = (int)smsg.get_result();
36          r.setDeviceID(new Integer(nodeid).toString());
37          r.setResult(new Integer(result));
38          devices.addResponseToDevice(nodeid, r);    }
39    } }
40
41    public void sendMessage(String nodeid, char message_type, String
        content){
42      SmartDeviceMsg smsg = new SmartDeviceMsg();
43      smsg.set_nodeid(gateway_tinyOS_id);
44      smsg.set_subject((short) message_type);
45
46      switch(message_type){
47        case 'H':  // Hello reply to device
48    case 'A':{ // Ack message to sensor mote
49          synchronized(mote){
50            mote.send(Integer.parseInt(nodeid), smsg); }
51    } } }
52
53    public void sendMessage(String nodeid, char message_type, Request
        req){
54      TinyOSParser parser = new TinyOSParser();
```

```
55      SmartDeviceMsg smsg = new SmartDeviceMsg();
56      smsg.set_nodeid(gateway_tinyOS_id);
57      smsg.set_subject((short) message_type);
58      String content = parser.parseRequestData(req);
59
60      char[] cdata = content.toCharArray();
61      short[] sdata = new short[cdata.length];
62      for(int i=0; i< cdata.length; i++)
63        sdata[i] = (short) cdata[i];
64      smsg.set_data(sdata);
65      synchronized(mote){
66        mote.send(Integer.parseInt(nodeid), smsg);
67    } } }
```

Listing E.2: Example libraries for communicating with Ploggs smart power outlets.

```
1  public class PloggMonitor extends Driver implements
       SerialPortEventListener, Runnable{
2
3    String comPort; // where the USB stick is located
4    OutputStream  out;
5          ParseBuffer   ps = new ParseBuffer();
6
7    public void startDevice (){
8      connect(comPort);  // not shown here
9      discoverDevices(); // not shown here
10   }
11
```

```
12  public void messageReceived(int dest_addr, Message msg){
13
14          String cmd;
15          Set<String> st = ps.Hdev.keySet();
16          Iterator<String> itr = st.iterator();
17          while (itr.hasNext()){
18            String id = itr.next();
19            cmd = "AT+UCAST:"+id+"=sv\r\n";
20            this.out.write(cmd.getBytes());
21          }
22  }
23
24  public void sendMessage(String nodeid, char messageType, Request
        request){
25
26    String id = request.getDeviceID();
27        String cmd = "AT+UCAST:"+id+"=sv\r\n";
28        do{
29          this.out.write(cmd.getBytes());
30        } while (!ps.enddata);
31        String res = this.JSONprint(ps.getHash(), id); // not shown
            here
32        Response r  = new Response(id, request.getServiceName(),
            request.getCommand(), res);
33        devices.addResponseToDevice(id, r);
34      }
35  }
```

Appendix F

Synchronous/Asynchronous Translation

The following code shows some parts of the synchronous/asynchronous synchronization mechanism (see Section 4.2.3), which was installed on the application framework in order to map synchronous HTTP requests, coming from Web clients (home tenants) at the Web server module of the application framework, to asynchronous operations occurring at the device layer of the framework, involving the physical interaction with home devices for satisfying the client requests.

The code has been developed in Java and it is embedded inside the *Device* class, which is the class thread responsible for managing embedded home devices.

Listing F.1: Code for supporting synchronous/asynchronous operation at the application framework for smart homes

```
1 package deviceLayer;
2
3 public class Device extends Observable
4     implements XMLRepresentable, Runnable {
5
6   // locking stuff...
```

```
7   private static Long msgToken = new Long(1);
8
9   /**
10   * @return a token for the message id.
11   */
12  public static synchronized long getToken() {
13    synchronized (msgToken)
14      return msgToken++;
15  }
16
17  /**
18   * dispatch a response to the synchronizer.
19   * @param r the low level response.
20   */
21  public static synchronized void dispatchResponse(Response r) {
22    AsyncToSync lock = synchronizer.get(r.getRequestID());
23    if (null == lock)
24      return;
25
26    synchronized (lock) {
27      lock.setResponse(r);
28      lock.notifyAll();
29    }
30  }
31
32  /**
```

```
33   * helper class to synchronize the async communication to tinyos/
         ploggs
34   *
35   */
36  public class AsyncToSync {
37
38    /** my token. */
39    private final long token;
40
41    /** the response onto my request. */
42    private Response response = null;
43
44    /**
45     * constructor.
46     */
47    public AsyncToSync() {
48      token = Device.getToken();
49    }
50
51    /**
52     * @return my token.
53     */
54    public long getToken() {
55      return token;
56    }
57
58    /**
```

```
59      * @return the response
60      */
61     public Response getResponse() {
62       return response;
63     }
64
65     /**
66      * @param response the response to set
67      */
68     public void setResponse(Response response) {
69       this.response = response;
70     }
71   };
72
73   /** a hash map containing the synchronizer objects. */
74   private static Map<Long, AsyncToSync> synchronizer =
75     new ConcurrentHashMap<Long, AsyncToSync> ();
76
77   // end of synchronizing
78
79   /*  Handles responses from home devices by forwarding them to the
       appropriate Web Client who made the Request */
80   public void handleResponse(Response r){
81     System.out.println("Handling␣normal␣Response␣for␣service:"+r.
         getServiceName());
82     dispatchResponse(r);
83   }
```

```
84
85  /* adds Request r in Request Message Queue */
86  public void addRequest(Request r){
87    this.msgQueue.addRequestMessage(r);
88  }
89
90  public String handle(org.restlet.data.Response response,
91      org.restlet.data.Request request) {
92
93    ... CODE THAT WAS OMITTED ...
94
95    Request re = new Request(deviceID,resourceName,method,params,
          values, false,0);
96    Response r = waitSynchronous(re);
97    if (null == r) {
98      response.setEntity(new StringRepresentation(Constants.NACK));
99    } else {
100     log.debug(r.getResult());
101     response.setEntity(new StringRepresentation(r.getResult().
            toString()));
102   }
103   return null;
104 }
105
106
107 public Response waitSynchronous(Request request) {
108   // handle a request with lock wait...
```

```
109     AsyncToSync lock = new AsyncToSync();
110     synchronizer.put(lock.getToken(), lock);
111     try{
112       request.setRequestID(lock.getToken());
113       addRequest(request);
114       synchronized (lock) { lock.wait(); }
115     } catch (Exception e) {
116       e.printStackTrace();
117       synchronizer.remove(lock);
118     }
119     synchronizer.remove(lock.getToken());
120     return lock.getResponse();
121   }
122 }
```

Appendix G

Sensor Programming without REST

We list the code that needs to be written in Java without employing REST, to achieve the same result as the second physical mashup provided in Section 4.2.6, developed in shell scripting.

Listing G.1: Sensor programming in Java to implement a distributed rule on sensor motes.

```
try{
  String deviceID = "sensor8";
  URL url = new URL("http://[" + deviceID + "]/Temperature");
  BufferedReader in = new BufferedReader(new
             InputStreamReader(url.openStream()));
  String str;
  while ((str = in.readLine()) != null) {
    output += str;
  }
  in.close();
}
} catch (Exception e) {

```

```
}

char[] checkVal =output.toCharArray();
String temperature = "";

for(int i=0; i < checkVal.length; i++){
  if(Character.isDigit(checkVal[i]))
    temperature += checkVal[i];
      else
            // use only digits for temperature
}

if(temperature < 20){
  String payload ="";
  // Construct payload data
  for(int i=0; i < request.getParameters().size(); i++){
    if(i == 0)
      payload = URLEncoder.encode(request.getParameters().get(i), "UTF
          -8") + "=" + URLEncoder.encode((String)request.getValues().
          get(i), "UTF-8");
    else
      payload += "&" + URLEncoder.encode(request.getParameters().get(i)
          , "UTF-8") + "=" + URLEncoder.encode((String)request.
          getValues().get(i), "UTF-8");
  }
          deviceID = "sensor5";
  URL url = new URL("http://[" + deviceID + "]/Light");
```

```
37    URLConnection conn = url.openConnection();
38    conn.setDoOutput(true);
39    OutputStreamWriter wr = new OutputStreamWriter(conn.getOutputStream()
          );
40    wr.write(payload);
41    wr.flush();
42    BufferedReader rd = new BufferedReader(new
43              InputStreamReader(conn.getInputStream()));
44    String line;
45    while ((line = rd.readLine()) != null) {
46      output += line;
47    }
48             // process output
49    wr.close();
50    rd.close();
51  }
```

In the physical mashup in Section 4.2.6, only seven lines of code were needed to implement a distributed rule on sensor motes, for checking the temperature of the room and turning on some LED it this temperature is less than 20 degrees Celsius. However, in Listing G.1, 50 lines of code were used to perform the same task, employing two sensor motes.

In the scenario when the sensor motes are not IPv6-enabled, this distributed rule would need approximately 200 lines to be executed. This happens because the 6LoWPAN stack makes the following command possible:

URL url = new URL(”http://[” + deviceID + ”]/Temperature”);

If not, a Java application would need to include also a *TinyOS driver*, implementing the *MessageListener* interface. It would also need a *TinyOSParser* class, to parse the contents of the message to the appropriate form understood by the sensor mote. Additionally, it would need the *MoteIF* class, for communicating with the mote, through the serial-to-USB port. We attach some of the additional code needed below.

Listing G.2: Additional code in Java in case 6LoWPAN is not supported.

```
1  public void sendMessage(String nodeid, char message_type, String
       payload) throws IOException{
2    private MoteIF mote;
3    TinyOSParser parser = new TinyOSParser();
4    SmartDeviceMsg smsg = new SmartDeviceMsg();
5    smsg.set_nodeid(FRAMEWORK_ID);
6    smsg.set_subject((short) message_type);
7    String content = parser.parseRequestData(payload);
8    switch(message_type){
9      case 'R':{  // Service Request message
10           try {
11             char[] cdata = content.toCharArray();
12             short[] sdata = new short[cdata.length];
13             for(int i=0; i< cdata.length; i++)
14               sdata[i] = (short) cdata[i];
15             smsg.set_data(sdata);
16             synchronized(mote){
17               mote.send(Integer.parseInt(nodeid), smsg);
18             }
19           } catch (IOException e) {
```

```
20              System.err.println("Cannot␣send␣Service␣Request␣Message␣to␣
                    mote");
21              e.printStackTrace();
22            }
23            break;
24          }
25          default:{
26            System.err.println("Unknown␣message␣subject.␣Sending␣failed."
                  );
27          }
28    }
29  }
```

Appendix H

Programming Telosb Sensor Motes

For people who are not familiar with sensor programming, we list here all the necessary steps needed in order to install IPv6/6LoWPAN support on Telosb sensor devices, transforming them then into embedded Web servers. At first, one must install the latest version of TinyOS[1] (currently TinyOS 2.1.2). A Web page with simple instructions for installing TinyOS, as well some additional needed libraries is the following:

http://www.tinyos.net/tinyos-2.x/doc/html/install-tinyos.html

After the necessary installations of TinyOS, as well as of tools, libraries and drivers for TinyOS support, the software code for transforming sensor motes into embedded Web servers must be installed. There are two different TinyOS programs: one that should be installed on the sensor mote acting as the base station (forwarding requests between the application framework and the remote sensor devices), and another that must be installed on the sensor motes which would be deployed in the smart home environment to sense its environmental context.

The software for an IPv6-enabled base station in TinyOS 2.x, can be found under the directory:

[1] http://www.tinyos.net/

```
{TinyOS Root Directory}/apps/IPBaseStation
```

To install the code on the sensor mote acting as the base station, a user must open a terminal and type the command:

```
make platform blip install.1 bsl,/dev/ttyUSBx
```

where *x* is the USB port where the mote is plugged in, *platform* is the sensor device type being used (e.g. telosb) and *blip* denotes that Blip[2] libraries will be used. The number after the *install* keyword, in this case *1*, identifies the address of the base station. The assigned USB port for each sensor mote can be found by typing the commad *motelist*. This command lists all the motes connected to the USB ports of the system.

If the following information appear on the terminal screen, this means that the TinyOS code has been successfully installed on the sensor mote.

```
    installing telosb binary using bsl
tos-bsl --telosb -c /dev/ttyUSB0 -r -e -I -p build/telosb/main.ihex.out-1
MSP430 Bootstrap Loader Version: 1.39-telos-8
Mass Erase...
Transmit default password ...
Invoking BSL...
Transmit default password ...
Current bootstrap loader version: 1.61 (Device ID: f16c)
Changing baudrate to 38400 ...
Program ...
20886 bytes programmed.
Reset device ...
rm -f build/telosb/main.exe.out-1 build/telosb/main.ihex.out-1
```

[2]http://smote.cs.berkeley.edu:8000/tracenv/wiki/blip

Afterwards, the base station needs to start its operation. This can be achieved through the following two successive commands:

```
cd {TinyOS Root Directory}/support/sdk/c/blip/

sudo driver/ip-driver /dev/ttyUSB0 telosb
```

Information similar to the following should appear on your terminal as soon as you run this command.

```
2012-08-16:22:19: INFO: Read config from 'serial_tun.conf'
2012-08-16:22:19: INFO: Using channel 15
2012-08-16:22:19: INFO: Retries: 1
2012-08-16:22:19: INFO: telnet console server running on port 6106
2012-08-16:22:19: INFO: created tun device: tun0
2012-08-16:22:19: INFO: interface device successfully initialized
2012-08-16:22:19: INFO: starting radvd on device tun0
```

Finally, all sensor motes, which will be deployed remotely in the smart home environment, need to install the TinyOS code we developed, for transforming them into embedded IPv6-enabled Web servers. The TinyOS source code for communicating with IPv6-enabled Telosb sensor motes (as well as Ploggs smart power outlets) has been released on the NetRL website[3] and on GitHub[4]. In order to install the code on a sensor mote, the following command must be executed:

```
make platform blip install.y bsl,/dev/ttyUSBx
```

Remember to use a different *y* number for each sensor mote, avoiding the value *1*, which is reserved for the base station node. As before, *x* is the USB port onto which the mote is temporarily plugged in (for the installation procedure).

[3] http://www.netrl.cs.ucy.ac.cy/index.php?option=com_content&task=view&id=76&Itemid=54

[4] https://github.com/andreaskami/homeweb

This is all! Now you can equip the sensor motes with batteries and deploy them inside the smart home. They will start immediately to scan the environment for an application framework (see the device discovery procedure in Section 5.2.1) and, as soon as they are bound to one, they will begin sensing the environmental conditions that exist in the house area, namely temperature, humidity and illumination.

Appendix I

Example WADL Service Description Files

Two example WADL service description files are presented, one for Telos sensor motes and one for Ploggs smart power outlets. The former case is shown in Listing I.1 and the latter in Listing I.2.

Listing I.1: An example WADL file for a Telosb sensor mote.

```
<?xml version="1.0"?>
<application xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             xmlns:html="http://www.w3.org/1999/xhtml"
             xmlns="http://wadl.dev.java.net/2009/02">
 <grammars></grammars>
 <resources base="http://localhost:8080/tinyOS/">
    <resource path="Temperature">
        <doc xml:lang="en" title="tinyOS Temperature Service">
           The <html:i>tinyOS Client Code</html:i> provides Services
               for motes' Temperature sensing capabilities.
        </doc>
        <method name="GET" id="Temperature">
```

```
12          <doc xml:lang="en" title="Measure Temperature in Celsius"/>
13          <request>
14          </request>
15          <response status="200">
16             <representation mediaType="text/plain" type="xsd:string">
17          <doc xml:lang="en" title="The Temperature sensed current
                value"/>
18               </representation>
19          </response>
20    </method>
21    </resource>
22    <resource path="Humidity">
23       <doc xml:lang="en" title="tinyOS Humidity Service">
24          The <html:i>tinyOS Client Code</html:i> provides Services for
                motes' Humidity sensing capabilities.
25       </doc>
26       <method name="GET" id="Humidity">
27           <doc xml:lang="en" title="Measure Humidity in % value"/>
28           <request>
29           </request>
30           <response status="200">
31              <representation mediaType="text/plain" type="xsd:string">
32              <doc xml:lang="en" title="The Humidity sensed current
                   value"/>
33              </representation>
34           </response>
35        </method>
```

```
36      </resource>
37      <resource path="Illumination">
38         <doc xml:lang="en" title="tinyOS Illumination Service">
39          The <html:i>tinyOS Client Code</html:i> provides services for
               motes' Radiation sensing capabilities.
40         </doc>
41         <method name="GET" id="Illumination">
42            <doc xml:lang="en" title="Measure Illumination in Lux value
                  "/>
43            <request>
44            </request>
45            <response status="200">
46               <representation mediaType="text/plain" type="xsd:string">
47               <doc xml:lang="en" title="The Illumination sensed current
                     value"/>
48               </representation>
49            </response>
50         </method>
51      </resource>
52      <resource path="Light">
53         <doc xml:lang="en" title="tinyOS Light Service">
54          The <html:i>tinyOS Client Code</html:i> provides Services
               setting motes' Leds.
55         </doc>
56         <method name="POST" id="Light">
57           <doc xml:lang="en" title="Set the RED/GREEN/BLUE Leds that are
                  on the sensor mote ON/OFF"/>
```

```
58          <request>
59               <param name="color" type="xsd:character" required="true"
                     default="" style="query">
60               <doc xml:lang="en" title="Leds Color to switch on"/>
61              <option value="R"/><option value="G"/><option value="B"/>
62               </param>
63         </request>
64         <response status="200">
65              <representation mediaType="text/plain" type="xsd:string">
66              <doc xml:lang="en" title="Acknowlegment Value"/>
67              </representation>
68         </response>
69        </method>
70     </resource>
71  </resources>
72 </application>
```

Listing I.2: An example WADL file for a Plogg smart power outlet.

```
1  <?xml version="1.0"?>
2  <application xmlns:xsd="http://www.w3.org/2001/XMLSchema"
3               xmlns:html="http://www.w3.org/1999/xhtml"
4               xmlns="http://wadl.dev.java.net/2009/02">
5   <grammars></grammars>
6   <resources base="http://localhost:8080/plogg/">
7      <resource path="Electricity">
8        <doc xml:lang="en" title="Plogg Smart Meter Electricity Service">
```

```
9         The <html:i>Electricity service</html:i> provides energy
              consumption measurements in Watts and kWh.
10      </doc>
11      <method name="GET" id="Electricity">
12      <doc xml:lang="en" title="Measure electrical consumption in Watts
            and kWh"/>
13      <request>
14      </request>
15      <response status="200">
16       <representation mediaType="application/json" type="xsd:string">
17          <doc xml:lang="en" title="The JSON value of the consumption
                measurement of the connected electrical appliance"/>
18        </representation>
19      </response>
20    </method>
21    </resource>
22    <resource path="Switch">
23      <doc xml:lang="en" title="Plogg Smart Meter Switch Service">
24        The <html:i>Switch service</html:i>turns an electrical
              appliance on/off.
25      </doc>
26      <method name="POST" id="Switch">
27         <doc xml:lang="en" title="Set the appliance ON/OFF"/>
28         <request>
29            <param name="mode" type="xsd:string" required="true"
                  default="" style="query">
```

```
                <doc xml:lang="en" title="Switch on/off the appliance
                    "/>
          <option value="On"/><option value="Off"/>
           </param>
         </request>
         <response status="200">
              <representation mediaType="text/plain" type="xsd:string">
              <doc xml:lang="en" title="Acknowlegment Value"/>
              </representation>
         </response>
      </method>
    </resource>
 </resources>
</application>
```

Appendix J

Important Factors affecting Domestic Energy Consumption

After discussions with the Electricity Authority of Cyprus, electricians and providers of smart home solutions, as well as tens of consumers of electricity in houses around Cyprus, we considered the following parameters, as important factors that affect the electrical energy consumption in domestic premises around the country:

- **Premise space**. The total physical area of the premise, in square meters.
- **Number of Adults**. The total number of adults living in the premise.
- **Number of Children**. The total number of children living in the premise.
- **Number of Rooms**. The total number of rooms in the premise.
- **Area**. Whether the premise s located at an urban/rural area or at the suburb of a city.
- **Residence Status**. Whether the premise is owned or rented.
- **Type of House**. Whether it is a house or an apartment/flat.
- **Swimming Pool**. Whether the premise has a swimming pool.

- **Jacuzzi/SPA**. Whether the premise has a jacuzzi or a spa.
- **Insulated**. Whether the premise is insulated or not.
- **Heating Type**. Specifies the technique employed to heat the premise. For example, it can be a central heating, halogen/gas heaters, a floor heating system, a fireplace etc.
- **Cooling Type**. Specifies the technique used to cool the premise. For example, it may be by means of air conditioners, fans etc.

There exist of course numerous other significant parameters (e.g. the confort levels of people, the time each tenant spends in his house), however, our intend is neither to complicate the collection procedure of this information from residents around Cyprus nor the data analysis process that would follow.

Bibliography

[1] LiteOS: A UNIX-like operating system for wireless sensor networks. Online at: http://www.liteos.net/ (accessed in December 2012).

[2] Nano-RK: A Wireless Sensor Networking Real-Time Operating System. Online at: http://www.nanork.org/projects/nanork/ (accessed in December 2012).

[3] Sen.se Sensor Platform. Online at: http://open.sen.se/ (accessed in December 2012).

[4] ThingSpeak. Online at: https://www.thingspeak.com/ (accessed in December 2012).

[5] WirelessHART Technology. Online at: http://www.hartcomm.org/protocol/wihart /wireless_technology.html (accessed in December 2012).

[6] Nielsen Online Report: Global Faces and Networked Places, March 2009. Online at: http://blog.nielsen.com/nielsenwire/global/social-networking-new-global-footprint/ (accessed in December 2012).

[7] Positive Energy Buildings thru Better Control Decisions (PEBBLE) FP7 European Project, January 2010. Online at: http://www.pebble-fp7.eu (accessed in December 2012).

[8] Smart Energy Efficiency Middleware for Public Spaces (SEEMPubS) FP7 European Project, December 2010. Online at: http://seempubs.polito.it/ (accessed in December 2012).

[9] Federal Information Processing Standards Publication 197. Specification for the Advanced Encryption Standard (AES), November 2001.

[10] Marco Aiello. The role of web services at home. In *Proceedings of the International Conference on Web Service-based Systems and Applications (WEBSA)*, page 164, Guadeloupe, French Caribbean, February 2006.

[11] Ian F. Akyildiz, W. Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, 2002.

[12] Hunt Allcott. Social norms and energy conservation. *Journal of Public Economics*, 95(9-10):1082–1095, October 2011.

[13] Gustavo Alonso, Fabio Casati, Harumi Kuno, and Vijay Machiraju. *Web Services: Concepts, Architectures*. Springer-Verlag, 2004.

[14] Paul M. Anderson, Abdel Aziz A. Fouad, Institute of Electrical, and Electronics Engineers. *Power system control and stability*. IEEE Press power engineering. 2003.

[15] Philip M. Anderson and M. Mirheydar. A low-order system frequency response model. *IEEE Transactions on Power Systems*, 5(3):720–729, August 1990.

[16] A. Aragues, I. Martinez, P. Del Valle, P. Munoz, J. Escayola, and J.D. Trigo. Trends in entertainment, home automation and e-health: Toward cross-domain integration. *IEEE Communications Magazine*, 50(6):160–167, June 2012.

[17] Jari Arko and Ari Keranen. CoAP Security Architecture, July 2011. IETF Draft, draft-arkko-core-security-arch-00.

[18] Antimo Barbato, Luca Borsani, Antonio Capone, and Stefano Melzi. Home energy saving through a user profiling system based on wireless sensors. In *First Workshop On Embedded Sensing Systems For Energy-Efficiency In Buildings (BuildSys)*, Berkeley, California, USA, November 2009. ACM.

[19] C. Beckel, H. Serfas, E. Zeeb, G. Moritz, F. Golatowski, and D. Timmermann. Requirements for smart home applications and realization with ws4d-pipesbox. In *Proceedings of the 16th Conference on Emerging Technologies Factory Automation (ETFA)*, pages 1–8, September 2011.

[20] Eric Bergman. *Information Appliances and Beyond*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000.

[21] Shah Bhatti, James Carlson, Hui Dai, Jing Deng, Jeff Rose, Anmol Sheth, Brian Shucker, Charles Gruenwald, Adam Torgerson, and Richard Han. MANTIS OS: An Embedded Multithreaded Operating System for Wireless Micro Sensor Platforms. *Mobile Networks and Applications*, 10(4):563–579, 2005.

[22] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[23] Tim Bray, Jean Paoli, C.M Sperberg-McQueen, Eve Maler, and Francois Yergeau. Extensible markup language (XML) 1.0 (fifth edition), 2008. Online at: http://www.w3.org/TR/REC-xml/ (accessed in December 2012).

[24] Barry Brumitt, Brian Meyers, John Krumm, Amanda Kern, and Steven Shafer. EasyLiving: Technologies for Intelligent Environments. In *Proceedings of the second International Symposium on Handheld and Ubiquitous Computing (HUC)*, pages 97–119, London, UK, 2000.

[25] Gabriella Castelli, Marco Mamei, and Franco Zambonelli. The changing role of pervasive middleware: From discovery and orchestration to recommendation and planning. In *Eighth International Workshop on Middleware and System Support for Pervasive Computing (PERWARE), at PerCom 2011*, pages 214–219, Seattle, USA, March 2011. IEEE.

[26] Nokia Research Center. Nokia Mobile Web Server. Online at: http://research.nokia.com/page/231 (accessed in December 2012).

[27] Shang-Wen Cheng, David Garlan, Bradley R. Schmerl, João Pedro Sousa, Bridget Spitznagel, Peter Steenkiste, and Ningning Hu. Software Architecture-Based Adaptation for Pervasive Systems. In *Proceedings of the International Conference on Architecture of Computing Systems (ARCS)*, pages 67–82, Karlsruhe, Germany, April 2002.

[28] Stuart Cheshire and Marc Krochmal. DNS-Based Service Discovery, December 2011. IETF Draft, draft-cheshire-dnsext-dns-sd.txt.

[29] Stuart Cheshire and Marc Krochmal. Multicast DNS, December 2011. IETF Draft, draft-cheshire-dnsext-multicastdns.txt.

[30] Stuart Cheshire, Marc Krochmal, and Kiren Sekar. Dynamic DNS Update Leases, August 2006. IETF Draft, draft-sekar-dns-ul-01.txt.

[31] Robert B. Cialdini. *Influence: science and practice*. Allyn and Bacon, 2001.

[32] Gabe Cohn, Sidhant Gupta, Jon Froehlich, Eric Larson, and Shwetak N. Patel. GasSense: Appliance-Level, Single-Point Sensing of Gas Activity in the Home. In *Proceedings of the eighth International Conference on Pervasive Computing (Pervasive)*, pages 265–282, Helsinki, Finland, May 2010.

[33] Diane J. Cook and Sajal K. Das. How smart are our environments? An updated look at the state of the art. *Pervasive and Mobile Computing*, 3(2):53–73, 2007.

[34] Robert B. Cooper. *Introduction to Queueing Theory*. North-Holland (Elsevier), second edition, 1981.

[35] D. Craciun, S. Ichim, and Y. Besanger. A new soft load shedding: Power system stability with contribution from consumers. In *Proceedings of PES PowerTech*, pages 1–6, Bucharest, Romania, July 2009.

[36] William C. Craig. Zigbee: Wireless Control That Simply Works. White paper, ZigBee Alliance.

[37] Sarah Darby. The effectiveness of feedback on energy consumption A review for defra of the literature on metering, billing and direct displays. *Environmental Change Institute, University of Oxford*, 2006.

[38] Stephen Dawson-Haggerty. Berkeley Blip 2.0: IP implementation for low-power networks. Online at: http://smote.cs.berkeley.edu:8000/tracenv/wiki/blip (accessed in December 2012).

[39] Stephen E. Deering and Robert M. Hinden. Internet Protocol Version 6 (IPv6) Specification, December 1998. RFC 2460.

[40] Mario Di Francesco, Sajal K. Das, and Giuseppe Anastasi. Data Collection in Wireless Sensor Networks with Mobile Elements: A Survey. *ACM Transactions on Sensor Networks (TOSN)*, 8(1):7:1–7:31, August 2011.

[41] Tim Dierks and Eric Rescorla. The Transport Layer Security (TLS) protocol Version 1.2. RFC 5246.

[42] Faiyaz Doctor, Hani Hagras, and Victor Callaghan. A fuzzy embedded agent-based approach for realizing ambient intelligence in intelligent inhabited environments. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 35(1):55–65, January 2005.

[43] Markus Weiss Dominique Guinard and Vlad Trifa. Energie Visible. Online at: http://www.webofthings.com/energievisible/ (accessed in December 2012).

[44] Vlad Trifa Dominique Guinard, Mathias Fischer. Sharing Using Social Networks in a Composable Web of Things. In *First International Workshop on the Web of Things (WoT), at PerCom 2010*, Mannheim, Germany, March 2010.

[45] Witold Drytkiewicz, Ilja Radusch, Stefan Arbanowski, and Radu Popescu-Zeletin. pREST: a REST-based protocol for pervasive systems. In *Proceedings of the first International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, pages 340–348, Fort Lauderdale, Florida, USA, October 2004.

[46] Adam Dunkels. Rime: A Lightweight Layered Communication Stack for Sensor Networks. In *Proceedings of the fourth European Conference on Wireless Sensor Networks (EWSN), Poster/Demo Session*, Delft, The Netherlands, January 2007.

[47] Adam Dunkels, Bjrn Grnvall, and Thiemo Voigt. Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors. In *First Workshop on Embedded Networked Sensors (EmNets)*, Tampa, Florida, USA, November 2004.

[48] Adam Dunkels, Oliver Schmidt, Thiemo Voigt, and Muneeb Ali. Protothreads: Simplifying Event-Driven Programming of Memory-Constrained Embedded Systems. In *Proceedings of the fourth Conference on Embedded Networked Sensor Systems (SenSys)*, Boulder, Colorado, USA, November 2006.

[49] Adam Dunkels, Thiemo Voigt, and Juan Alonso. Making TCP/IP Viable for Wireless Sensor Networks. In *Proceedings of the first European Workshop on Wireless Sensor Networks (EWSN), work-in-progress session*, Berlin, Germany, January 2004.

[50] Mathilde Durvy, Julien Abeillé, Patrick Wetterwald, Colin O'Flynn, Blake Leverett, Eric Gnoske, Michael Vidales, Geoff Mulligan, Nicolas Tsiftes, Niclas Finne, and Adam Dunkels. Making Sensor Networks IPv6 Ready. In *Proceedings of the Sixth Conference on Networked Embedded Sensor Systems (SenSys), Poster Session*, Raleigh, North Carolina, USA, November 2008.

[51] Karen Ehrhardt-Martinez, Kat Donnelly, and John A. "Skip" Laitner. Advanced Metering Initiatives and Residential Feedback Programs: A Meta-Review for Household Electricity-Saving Opportunities, 2010. ACEEE Report No. E105.

[52] Jarno Elonen. NanoHTTPD embeddable HTTP server in Java. Online at: http://elonen.iki.fi/code/nanohttpd/ (accessed in December 2012).

[53] Christoph Endres, Andreas Butz, and Asa MacWilliams. A survey of software infrastructures and frameworks for ubiquitous computing. *Mobile Information Systems*, 1(1):41–80, 2005.

[54] Sinem Coleri Ergen. ZigBee/IEEE 802.15.4 Summary, December 2004. Online at: http://www.sinemergen.com/zigbee.pdf (accessed in December 2012).

[55] Varick L. Erickson, Yiqing Lin, Ankur Kamthe, Rohini Brahme, Amit Surana, Alberto E. Cerpa, Michael D. Sohn, and Satish Narayanan. Energy efficient building environment control strategies using real-time occupancy measurements. In *First Workshop On Embedded Sensing Systems For Energy-Efficiency In Buildings (BuildSys)*, Berkeley, California, USA, November 2009.

[56] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys*, 35(2):114–131, June 2003.

[57] Europa Press Release. Communication from the European Commission. Energy Efficiency: Delivering the 20% target. November 2008.

[58] Jay Farrell and Marios Polycarpou. *Adaptive approximation based control: unifying neural, fuzzy and traditional adaptive approximation approaches*. Adaptive and learning systems for signal processing, communications, and control. Wiley-Interscience, 2006.

[59] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, Irvine, California, 2000.

[60] Michael Franklin and Stan Zdonik. Data in your face: push technology in perspective. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 516–519, Seattle, Washington, USA, June 1998.

[61] Jon E. Froehlich, Eric Larson, Tim Campbell, Conor Haggerty, James Fogarty, and Shwetak N. Patel. Hydrosense: infrastructure-mediated single-point sensing of whole-home water activity. In *Proceedings of the 11th International Conference on Ubiquitous Computing (UbiComp)*, pages 235–244, Orlando, Florida, USA, December 2009.

[62] Neil Gershenfeld, Raffi Krikorian, and Danny Cohen. The Internet of Things. *Scientific American*, 291(4):76–81, October 2004.

[63] Phillip Gibbons, Brad Karp, Yan Ke, Suman Nath, and Srinivasan Seshan. Irisnet: an architecture for a worldwide sensor web. *IEEE Pervasive Computing*, 2(4):22–33, October 2003.

[64] Yaron Y. Goland, Ting Cai, Paul Leach, and Ye Gu. Simple Service Discovery Protocol, October 1999. IETF Draft, draft-cai-ssdp-v1-03.txt.

[65] Carles Gomez and Josep Paradells. Wireless home automation networks: A survey of architectures and technologies. *IEEE Communications Magazine*, 48(6):92–101, 2010.

[66] Google. PowerMeter Project. Online at: http://www.google.com/powermeter/about/ (accessed in December 2012).

[67] Joe Gregorio and Bill de hOra. The Atom Publishing Protocol, October 2007. RFC 5023.

[68] Christin Groba and Siobhan Clarke. Web services on embedded systems - a performance study. In *First International Workshop on the Web of Things (WoT), at PerCom 2010*, pages 726–731, Mannheim, Germany, March 2010.

[69] Network Working Group. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, 1996. Online at: http://tools.ietf.org/html/rfc2045 (accessed in December 2012).

[70] Network Working Group. The application/json Media Type for JavaScript Object Notation (JSON), 2006. Online at: http://tools.ietf.org/html/rfc4627/ (accessed in December 2012).

[71] Dominique Guinard. Mashing up Your Web-Enabled Home. In *First International Workshop on Web-Enabled Objects (TouchTheWeb), at ICWE 2010*, Vienna, Austria, July 2010.

[72] Dominique Guinard. *A Web of Things Application Architecture - Integrating the Real-World into the Web*. PhD thesis, ETH Zurich, Zurich, Switzerland, August 2011.

[73] Dominique Guinard, Iulia Ion, and Simon Mayer. In Search of an Internet of Things Service Architecture: REST or WS-*? A Developers Perspective. In *Proceedings of the eighth International ICST Conference on Mobile and Ubiquitous Systems (Mobiquitous)*, Copenhagen, Denmark, December 2011.

[74] Dominique Guinard, Mathias Mueller, and Jacques Pasquier-Rocha. Giving RFID a REST: Building a Web-Enabled EPCIS. In *Procedings of the second International Conference on the Internet of Things (IoT)*, Tokyo, Japan, December 2010.

[75] Dominique Guinard and Vlad Trifa. Towards the Web of Things: Web Mashups for Embedded Devices. In *Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM), at WWW 2009*, Madrid, Spain, April 2009.

[76] Dominique Guinard, Vlad Trifa, and Erik Wilde. Architecting a Mashable Open World Wide Web of Things. Technical Report No. 663, Dept. of Computer Science, ETH Zurich, February 2010.

[77] Arnt Gulbrandsen, Paul Vixie, and Levon Esibov. A DNS RR for specifying the location of services, February 2000. RFC 2782.

[78] Manu Gupta, Stephen S. Intille, and Kent Larson. Adding GPS-Control to Traditional Thermostats: An Exploration of Potential Energy Savings and Design Challenges. In *Proceedings of the seventh International Conference on Pervasive Computing (Pervasive)*, pages 95–114, Nara, Japan, May 2009.

[79] Sidhant Gupta, Matthew S. Reynolds, and Shwetak N. Patel. ElectriSense: single-point sensing using EMI for electrical event detection and classification in the home. In *Proceedings of the 12th International Conference on Ubiquitous Computing (Ubicomp)*, pages 139–148, Copenhagen, Denmark, December 2010.

[80] Erik Guttman, Charles Perkins, John Veizades, and Michael Day. Service Location Protocol, Version 2, June 1999. RFC 2608.

[81] Darrel Hankerson, Alfred J. Menezes, and Scott Vanstone. *Guide to Elliptic Curve Cryptography*. Springer-Verlag, Secaucus, USA, 2003.

[82] Usman Haque. Cosm - Internet of Things Platform. Online at: https://cosm.com/ (accessed in December 2012).

[83] Colin Harris and Vinny Cahill. An empirical study of the potential for context-aware power management. In *Proceedings of the ninth International Conference on Ubiquitous Computing (UbiComp)*, pages 235–252, Innsbruck, Austria, December 2007. Springer.

[84] Sumi Helal, William Mann, Hicham El-Zabadani, Jeffrey King, Youssef Kaddoura, and Erwin Jansen. The Gator Tech Smart House: A Programmable Pervasive Space. *Computer*, 38(3):50–60, March 2005.

[85] Yu Huang, Xiaoxing Ma, Jiannong Cao, Xianping Tao, and Jian Lu. Concurrent Event Detection for Asynchronous consistency checking of pervasive context. In *Proceedings of the International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–9, Galveston, Texas, USA, March 2009.

[86] Jonathan W. Hui and David E. Culler. IP is dead, long live IP for wireless sensor networks. In *Proceedings of the sixth Conference on Networked Embedded Sensor Systems (SenSys)*, pages 15–28, Raleigh, NC, USA, November 2008.

[87] Urs Hunkeler, Hong Linh Truong, and Andy Stanford-Clark. MQTT-S - A publish/subscribe protocol for Wireless Sensor Networks. In *Proceedings of the third International Conference on COMmunication System softWAre and MiddlewaRE (COMSWARE)*, pages 791–798, Bangalore, India, January 2008.

[88] Myunggwon Hwang, Dongjin Choi, and Pankoo Kim. Least Slack Time Rate first: an Efficient Scheduling Algorithm for Pervasive Computing Environment. *Journal of Universal Computer Science)*, 17(6):912–925, 2011.

[89] International Energy Agency. World Energy Outlook. 2007.

[90] Stephen S. Intille. Designing a Home of the Future. *IEEE Pervasive Computing*, 1(2):76–82, 2002.

[91] Giulio Jacucci, Anna Spagnolli, Luciano Gamberini, Alessandro Chalambalakis, Christoffer Bjorkskog, Massimo Bertoncini, Carin Torstensson, and Pasquale Monti. Designing Effective feedback of Electricity Consumption for Mobile User Interfaces. *PsychNology Journal*, 7(3):265–289, 2009.

[92] Marco Jahn, Marc Jentsch, Christian R. Prause, Ferry Pramudianto, and Amro Al-Akkad andRen Reiners. The Energy Aware Smart Home. *Proceedings of the fifth International Conference on Future Information Technology (FutureTech)*, pages 1–8, May 2010.

[93] Xiaofan Jiang, Stephen Dawson-Haggerty, Prabal Dutta, and David Culler. Design and implementation of a high-fidelity AC metering network. In *Proceedings of the eighth International Conference on Information Processing in Sensor Networks (IPSN)*, pages 253–264, Washington, DC, USA, April 2009.

[94] Anders Wallberg Par Hansson Olov Stahl Jonas Soderberg Jussi Karlgren, Lennart E. Fahlen and Karl-Petter Akesson. Socially intelligent interfaces for increased energy awareness in the home. In *Proceedings of the first International Conference on the Internet of Things (IoT)*, pages 263–275, Zurich, Switzerland, 2008.

[95] Andreas Kamilaris, Nicolas Iannarilli, Vlad Trifa, and Andreas Pitsillides. Bridging the Mobile Web and the Web of Things in Urban Environments. In *First International Workshop on the Urban Internet of Things (Urban IoT), at IoT 2010*, Tokyo, Japan, November 2010.

[96] Andreas Kamilaris and Andreas Pitsillides. Using DNS for Global Discovery of Environmental Services. In *Proceedings of the eighth International Conference on Web Information Systems and Technologies (WEBIST)*, Porto, Portugal, April 2012.

[97] Kat A. Donnelly Karen Ehrhardt-Martinez and John A. Laitner. Advanced Metering Initiatives and Residential Feedback Programs: A Meta-Review for Household Electricity-Saving Opportunities, Research Report E105. *American Council for an Energy-Efficient Economy*, June 2010. Online at: http://aceee.org/research-report/e105 (accessed in December 2012).

[98] Stephan Karpischek and Florian Michahelles. my2cents - Digitizing consumer opinions and comments about retail products. In *Procedings of the second International Conference on the Internet of Things (IoT)*, Tokyo, Japan, December 2010.

[99] Stephen Kent and Karen Seo. Security Architecture for the Internet Protocol. RFC 4301.

[100] Fotis Kerasiotis, Aggeliki Prayati, Christos Antonopoulos, Christos Koulamas, and George Papadopoulos. Battery Lifetime Prediction Model for a WSN Platform. In *Proceedings of the fourth International Conference on Sensor Technologies and Applications (SENSORCOMM)*, pages 525–530, Venice/Mestre, Italy, July 2010.

[101] Cory D. Kidd, Robert Orr, Gregory D. Abowd, Christopher G. Atkeson, Irfan A. Essa, Blair MacIntyre, Elizabeth D. Mynatt, Thad Starner, and Wendy Newstetter. The Aware Home: A Living Laboratory for Ubiquitous Computing Research. In *Second International Workshop on Cooperative Buildings, Integrating Information, Organization, and Architecture (CoBuild)*, pages 191–198, 1999.

[102] Younghun Kim, Thomas Schmid, Zainul M. Charbiwala, and Mani B. Srivastava. ViridiScope: design and implementation of a fine grained power monitoring system for homes. In *Proceedings of the 11th International Conference on Ubiquitous Computing (Ubicomp)*, pages 245–254, Orlando, Florida, USA, December 2009.

[103] Tim Kindberg, John Barton, Jeff Morgan, Gene Becker, Debbie Caswell, and Philippe Debaty. People, places, things: web presence for the real world. *Mobile Networks and Applications*, 7(5):365–376, October 2002.

[104] George J. Klir and Bo Yuan. *Fuzzy sets and fuzzy logic: theory and applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1995.

[105] Matthias Kovatsch, Markus Weiss, and Dominique Guinard. Embedding Internet Technology for Home Automation. In *Proceedings of the 15th Conference on Emerging Technologies and Factory Automation (ETFA)*, Bilbao, Spain, December 2010.

[106] Vinay Kumar and Sudarshan Tiwari. Routing in IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN): A Survey. *Journal of Computer Networks and Communications*, 2012, 2012.

[107] Nandakishore Kushalnagar, Gabriel Montenegro, and Christian Schumacher. IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals, August 2007. RFC 4919.

[108] Stacey Kuznetsov and Eric Paulos. Participatory Sensing in Public Spaces: Activating Urban Surfaces with Sensor Probes. In *Proceedings of the eighth Conference on Designing Interactive Systems (DIS)*, Aarhus, Denmark, August 2010.

[109] Victor Lesser, Michael Atighetchi, Brett Benyo, Bryan Horling, Victor Lesser Michael Atighetchi, Anita Raja, Regis Vincent, Ping Xuan, Shelley XQ. Zhang, Thomas Wagner, Ping Xuan, and Shelley Xq. Zhang. The intelligent home testbed. In *Workshop on Autonomy Control Software*, May 1999.

[110] Philip Levis, Sam Madden, Joseph Polastre, Robert Szewczyk, Alec Woo, David Gay, Jason Hill, Matt Welsh, Eric Brewer, and David Culler. TinyOS: An operating system for sensor networks. *Ambient Intelligence*, pages 115–148, 2004.

[111] Lassi A. Liikanen and Tatu Nieminen. Comparison of End User Electric Power Meters for Accuracy. Technical report, Helsinki Institute for Information Technology (HIIT), 2009.

[112] Thomas Luckenbach, Peter Gober, Stefan Arbanowski, Andreas Kotsopoulos, and Kyle Kim. TinyREST - a protocol for integrating sensor networks into the internet. Stockholm, Sweden, June 2005.

[113] Johnny Rodgers Lyn Bartram and Rob Woodbury. Supporting Sustainable Living: Aware Homes and Smart Occupants. In *Workshop on Ubiquitous Computing for Sustainable Energy (UCSE), at UbiComp 2010*, Copenhagen, Denmark, December 2010.

[114] Alan Marchiori and Qi Han. Using Circuit-Level Power Measurements in Household Energy Management Systems. In *First Workshop On Embedded Sensing Systems For Energy-Efficiency In Buildings (BuildSys)*, Berkeley, California, USA, November 2009.

[115] Friedemann Mattern and Christian Floerkemeier. From the Internet of Computers to the Internet of Things. chapter From Active Data Management to Event-Based Systems and More, pages 242–259. Springer-Verlag, Berlin, Heidelberg, 2010.

[116] Friedemann Mattern, Thorsten Staake, and Markus Weiss. Ict for green: how computers can help us to conserve energy. In *Proceedings of the first International Conference on Energy-Efficient Computing and Networking (e-Energy)*, pages 1–10, Passau, Germany, April 2010.

[117] Microsoft. Hohm Project. Online at: http://www.microsoft-hohm.com/ (accessed in December 2012).

[118] Paul Mockapetris. Domain Names - Implementation and Specifications, November 1987. STD 13, RFC 1035.

[119] Gabriel Montenegro, Nandakishore Kushalnagar, Jonathan W. Hui, and David E. Culler. 6LoWPAN: Transmission of IPv6 Packets over IEEE 802.15.4 Networks, December 2007. RFC 4944.

[120] Michael C. Mozer. Smart environments: Technologies, protocols, and applications. chapter Lessons from an Adaptive Home, pages 271–294. John Wiley and Sons, Inc., Hoboken, NJ, USA, 2005.

[121] Julian Neil, Chris Wallace, and Kevin Korb. Learning Bayesian Networks with Restricted Causal Interactions. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 486–493, Stockholm, Sweden, July 1999.

[122] Donald A. Norman. *The Invisible Computer: Why Good Products Can Fail, the Personal Computer Is So Complex, and Information Appliances Are the Solution*, volume 1. MIT Press, 1 edition, 1999.

[123] Mark Nottingham and Robert Sayre. The Atom Syndication Format, December 2005. RFC 4287.

[124] The White House. Office of Science and Technology Policy. Empowering Customers With a Green Button. Online at: http://www.whitehouse.gov/blog/2011/11/21/empowering-customers-green-button (accessed in December 2012).

[125] Benedikt Ostermaier, Kay Roemer, Friedem Mattern, Michael Fahrmair, and Wolfgang Kellerer. A Real-Time Search Engine for the Web of Things. In *Proceedings of the second International Conference on the Internet of Things (IoT)*, Tokyo, Japan, December 2010.

[126] Benedikt Ostermaier, Kay Roemer, Friedem Mattern, Michael Fahrmair, and Wolfgang Kellerer. A Real-Time Search Engine for the Web of Things. In *Proceedings of the International Conference on the Internet of Things (IoT)*, Tokyo, Japan, December 2010.

[127] Benedikt Ostermaier, Fabian Schlup, and Kay Römer. WebPlug: A framework for the Web of Things. In *First International Workshop on the Web of Things (WoT), at PerCom 2010*, Mannheim, Germany, 2010.

[128] Filippos Papadopoulos, Apostolos Zarras, Evaggelia Pitoura, and Panos Vassiliadis. Timely provisioning of mobile services in critical pervasive environments. In *Proceedings of the Confederated International Conference on the Move to Meaningful Internet Systems (OTM)*, pages 864–881, Agia Napa, Cyprus, October 2005.

[129] Florence Passy and Marco Giugni. Social Networks and Individual Perceptions: Explaining Differential Participation in Social Movements. *Sociological Forum*, 16(1):123–153, 2001.

[130] Cesare Pautasso, Olaf Zimmermann, and Frank Leymann. RESTful Web Services vs. "big" web services: making the Right Architectural Decision. In *Proceedings of the 17th International World Wide Web Conference (WWW)*, pages 805–814, Beijing, China, April 2008.

[131] Alfredo J. Perez, Miguel A. Labrador, and Sean J. Barbeau. G-sense: a scalable architecture for global sensing and monitoring. *IEEE Network*, 24(4):57–64, July 2010.

[132] Dane Petersen, Jay Steele, and Joe Wilkerson. WattBot: a residential electricity monitoring and feedback system. In *Proceedings of the 27th International Conference on Human factors in Computing Systems (CHI)*, pages 2847–2852, Boston, MA, USA, April 2009.

[133] Arun G. Phadke and Bogdan Kasztenny. Synchronized Phasor and Frequency Measurement Under Transient Conditions. *IEEE Transactions on Power Delivery*, 24(1):89–95, January 2009.

[134] Joseph Polastre, Robert Szewczyk, and David Culler. Telos: enabling ultra-low power wireless research. In *Proceedings of the fourth International Symposium on Information Processing in Sensor Networks (IPSN)*, Los Angeles, California, USA, April 2005. IEEE Press.

[135] Christian Prehofer, Jilles van Gurp, and Cristiano di Flora. Towards the web as a platform for ubiquitous applications in smart spaces. In *Second Workshop on Requirements and Solutions for Pervasive Software Infrastructures (RSPSI), at Ubicomp 2007*, Innsbruck, Austria, December 2007.

[136] Christian Prehofer, Jilles van Gurp, Vlad Stirbu, Sailesh Satish, Sasu Tarkoma, Cristiano di Flora, and Pasi P. Liimatainen. Practical Web-Based Smart Spaces. *IEEE Pervasive Computing*, 9(3):72–80, 2010.

[137] Nissanka B. Priyantha, Aman Kansal, Michel Goraczko, and Feng Zhao. Tiny web services: design and implementation of interoperable and evolvable sensor networks. In *Proceedings of the sixth Conference on Embedded Network Sensor Systems (SenSys)*, pages 253–266, Raleigh, NC, USA, November 2008.

[138] W. Fred Van Raaij and Theo M. M. Verhallen. A behavioral model of residential energy use. *Journal of Economic Psychology*, 3(1):39–63, 1983.

[139] Yakov Rekhter, Susan Thomson, Jim Bound, and Paul Vixie. Dynamic Updates in the Domain Name System (DNS UPDATE), April 1997. RFC 2136.

[140] Eric Rescorla. HTTP over TLS. RFC 2818.

[141] Eric Rescorla and Nagendra Modadug. Datagram Transport Layer Security. RFC 4347.

[142] Leonard Richardson and Sam Ruby. *RESTful Web Services*. O'Reilly, 2007.

[143] Stephen E. Deering Robert M. Hinden. IP Version 6 Addressing Architecture, February 2006. RFC 4291.

[144] Manuel Romn, Christopher Hess, Renato Cerqueira, Roy H. Campbell, and Klara Nahrstedt. Gaia: A Middleware Infrastructure to Enable Active Spaces. *IEEE Pervasive Computing*, 1:74–83, 2002.

[145] Peter Saint-Andre. Extensible Messaging and Presence Protocol (XMPP): Core, June 2009. IETF Draft, draft-ietf-xmpp-3920bis-00.

[146] Andre Santanche, Suman Nath, Jie Liu, Bodhi Priyantha, and Feng Zhao. SenseWeb: Browsing the Physical World in Real Time. In *Proceedings of the fifth International Conference on Information Processing in Sensor Networks (IPSN)*, Nashville, TN, USA, April 2006.

[147] Lars Schor, Philipp Sommer, and Roger Wattenhofer. Towards a Zero-Configuration Wireless Sensor Network Architecture for Smart Buildings. In *First Workshop On Embedded Sensing Systems For Energy-Efficiency In Buildings (BuildSys)*, Berkeley, California, USA, November 2009.

[148] Peter Schramm, Edwin Naroska, Peter Resch, Jorg Platte, Holger Linde, Guido Stromberg, and Thomas Sturm. A service gateway for networked sensor systems. *IEEE Pervasive Computing*, 3(1):66–74, January 2004.

[149] Christian Seitz, Christoph Legat, and Jorg Neidig. Embedding Semantic Product Memories in the Web of Things. In *First International Workshop on the Web of Things (WoT), at PerCom 2010*, Mannheim, Germany, March 2010.

[150] Zach Shelby, Klaus Hartke, Carsten Bormann, and Brian Frank. Constrained Application Protocol (CoAP), March 2012. IETF Draft, draft-ietf-core-coap-09.

[151] Myung-Ki Shin, Tiago Camilo, Jorge Sa Silva, and Dominik Kaspar. Mobility Support in 6LoWPAN, November 2007. IETF Draft, draft-shin-6lowpan-mobility-01.

[152] Joao Pedro Sousa and David Garlan. Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments. In *Proceedings of the third Working IEEE/IFIP Conference on Software Architecture*, pages 29–43, Deventer, The Netherlands, 2002. Kluwer Academic Publishers.

[153] Vlad Stirbu. Towards a RESTful Plug and Play Experience in the Web of Things. In *Proceedings of the fifth International Conference on Semantic Computing (ICSC)*, pages 512–517, Santa Clara, CA, USA, August 2008. IEEE.

[154] Salman Taherian, Marcelo Pias, George Coulouris, and Jon Crowcroft. Profiling energy use in households and office spaces. In *Proceedings of the first International Conference on Energy-Efficient Computing and Networking (e-Energy)*, pages 21–30, Passau, Germany, April 2010.

[155] Emmanuel Munguia Tapia, Stephen S. Intille, and Kent Larson. Activity Recognition in the Home Using Simple and Ubiquitous Sensors. In *Proceedings of the second International Conference on Pervasive Computing (Pervasive)*, Vienna, Austria, April 2004.

[156] Ben L. Titzer, Daniel K. Lee, and Jens Palsberg. Avrora: scalable sensor network simulation with precise timing. In *Proceedings of the fourth International Symposium on Information Processing in Sensor Networks (IPSN)*, Los Angeles, California, USA, April 2005.

[157] Serbulent Tozlu, Murat Senel, Wei Mao, and Abtin Keshavarzian. Wi-Fi Enabled Sensors for Internet of Things: A Practical Ashpproach., volume = 50, year = 2012. *IEEE Communications Magazine*, (6):134–143.

[158] Bernard Traversat, Mohamed Abdelaziz, Dave Doolin, Mike Duigou, Jean-Christophe Hugly, and Eric Pouyoul. Project JXTA-C: enabling a Web of things. Big Island, Hawaii, USA, January 2003.

[159] Vlad Trifa. *Building Blocks for a Participatory Web of Things: Devices, Infrastructures, and Programming Frameworks*. PhD thesis, ETH Zurich, Zurich, Switzerland, August 2011.

[160] Vlad Trifa, Dominique Guinard, Philipp Bolliger, and Samuel Wieland. Design of a Web-based Distributed Location-Aware Infrastructure for Mobile Devices. In *First International Workshop on the Web of Things (WoT), at PerCom 2010*, Mannheim, Germany, March 2010.

[161] Vlad Trifa, Dominique Guinard, Vlatko Davidovski, Andreas Kamilaris, and Ivan Delchev. Web messaging for open and scalable distributed sensing applications. In *Proceedings of the 10th International Conference on Web Engineering (ICWE)*, pages 129–143, Vienna, Austria, July 2010.

[162] Yu-Chee Tseng, You-Chiun Wang, and Lun-Wu Yeh. iPower: an energy conservation system for intelligent buildings by wireless sensor networks. *International Journal of Sensor Networks*, 5(1):1–10, 2009.

[163] United Nations. Millennium Development Goals Report, 2007. Online at: http://www.un.org/millenniumgoals/pdf/mdg2007.pdf (accessed in December 2012).

[164] J. van Gurp, C. Prehofer, and C. di Flora. Experiences with Realizing Smart Space Web Service Applications. In *Proceedings of the fifth Consumer Communications and Networking Conference (CCNC)*, pages 1171–1175, Las Vegas, NV, USA, January 2008.

[165] Jeannet H Van Houwelingen and W Fred Van Raaij. The Effect of Goal-Setting and Daily Electronic Feedback on In-Home Energy Use. *Journal of Consumer Research*, 16(1):98–105, 1989.

[166] Juan Vazquez, Aitor Almeida, Iker Doamo, Xabier Laiseca, and Pablo Ordua. Flexeo: An architecture for integrating wireless sensor networks into the internet of things. In *Proceedings of the third Symposium of Ubiquitous Computing and Ambient Intelligence (UCAmI)*, volume 51 of *Advances in Soft Computing*, pages 219–228. Springer Berlin, Heidelberg, Salamanca, Spain, October 2009.

[167] Juan Vazquez, Diego de Ipia, and Iigo Sedano. SOAM: An Environment Adaptation Model for the Pervasive Semantic Web. In *Procedings of the International Conference on Computational Science and Its Applications (ICCSA)*, volume 3983 of *Lecture Notes in Computer Science*, pages 108–117, Glasgow, UK, May 2006. Springer Berlin, Heidelberg.

[168] Juan Ignacio Vazquez, Jonathan Ruiz de Garibay, Xabier Eguiluz, Iker Doamo, Silvia Rentera, and Ana Ayerbe. Communication Architectures and Experiences for Web-Connected Physical Smart Objects. In *First International Workshop on the Web of Things (WoT), at PerCom 2010*, Mannheim, Germany, 2010.

[169] Juan Ignacio Vazquez and Diego Lopez-De-Ipina. Social devices: autonomous artifacts that communicate on the internet. In *Proceedings of the first International Conference on the Internet of Things (IoT)*, pages 308–324, Zurich, Switzerland, March 2008.

[170] H.K. Jerry Chu Matt Sargent Vern Paxson, Mark Allman. Computing TCP's Retransmission Timer, June 2011. RFC 6298.

[171] Slobodanka Tomic Rene Mayrhofer Tassilo Pellegrini Vikash Kumar, Anna Fensel. User Created Machine-readable Policies for Energy Efficiency in Smart Homes. In *Workshop on Ubiquitous Computing for Sustainable Energy (UCSE), at UbiComp 2010*, Copenhagen, Denmark, December 2010.

[172] Haodong Wang, Chiu Chiang Tan, and Qun Li. Snoogle: A Search Engine for the Physical World. In *Proceedings of the 27th Conference on Computer Communications (INFOCOM)*, pages 1382–1390, Phoenix, AZ, USA, April 2008.

[173] Yu Wang and Hongyi Wu. Delay/Fault-Tolerant Mobile Sensor Network (DFT-MSN): A New Paradigm for Pervasive Information Gathering. *IEEE Transactions on Mobile Computing*, 6(9):1021–1034, 2007.

[174] Roy Want, Kenneth P. Fishkin, Anuj Gujar, and Beverly L. Harrison. Bridging physical and virtual worlds with electronic tags. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 370–377, Pittsburgh, Pennsylvania, USA, May 1999.

[175] Mark Weiser. The computer for the 21st century. *ACM SIGMOBILE Mobile Computing and Communications Review*, 3(3):3–11, 1999.

[176] Markus Weiss, Adrian Merkle, Thorsten Staake, and Elgar Fleisch. Towards a PowerPedia A Collaborative Energy Encyclopedia. In *Workshop on Ubiquitous Computing for Sustainable Energy (UCSE), at UbiComp 2010*, Copenhagen, Denmark, December 2010.

[177] Geoffrey Werner-Allen, Konrad Lorincz, Matt Welsh, Omar Marcillo, Jeff Johnson, Mario Ruiz, and Jonathan Lees. Deploying a Wireless Sensor Network on an Active Volcano. *IEEE Internet Computing*, 10(2):18–25, March 2006.

[178] Erik Wilde. Putting things to REST. Technical Report UCB iSchool Report 2007-015, School of Information, UC Berkeley, November 2007.

[179] G. Wood and M. Newborough. Dynamic energy-consumption indicators for domestic appliances: environment, behaviour and design. *Energy and Buildings*, 35(8):821–841, 2003.

[180] Chao-Lin Wu, Chun-Feng Liao, and Li-Chen Fu. Service-Oriented Smart-Home Architecture Based on OSGi and Mobile-Agent Technology. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 37(2):193–205, March 2007.

[181] Qinglong Wu, Fei-Yue Wang, and Yuetong Lin. A mobile-agent based distributed intelligent control system architecture for home automation. In *Proceedings of the International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1599–1605, Big Island, Hawaii, USA, October 2005.

[182] Dogan Yazar and Adam Dunkels. Efficient Application Integration in IP-based Sensor Networks. In *First Workshop On Embedded Sensing Systems For Energy-Efficiency In Buildings (BuildSys)*, Berkeley, California, November 2009.

[183] G. Michael Youngblood, Diane J. Cook, and Lawrence B. Holder. Managing Adaptive Versatile environments. *Pervasive and Mobile Computing*, 1(4):373–403, December 2005.

[184] Jerry Zhao and Ramesh Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *Proceedings of the first International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 1–13, Los Angeles, California, USA, 2003.