Architecture and Protocols for the Internet of Things: A Case Study

Angelo P. Castellani*†, Nicola Bui†, Paolo Casari*, Michele Rossi*, Zach Shelby‡, Michele Zorzi*†
*Department of Information Engineering, University of Padova, Via Gradenigo 6/B, I-35131 Padova, Italy
†Consorzio Ferrara Ricerche (CFR), Via Saragat 1, B building, I-44100 Ferrara, Italy
‡Sensinode Ltd., Hallituskatu 13-17 D, FIN-90100 Oulu, Finland
Email: {castellani,casarip,rossi}@dei.unipd.it, {bui,zorzi}@ing.unife.it, zach@sensinode.com

***Abstract*—In this paper, we describe a practical realization of an Internet-of-Things (IoT) architecture at the University of Padova, Italy. Our network spans the floors of different buildings within the Department of Information Engineering, and is designed to provide access to basic services such as environmental monitoring and localization to University users, as well as to manage service access based on user roles and authorizations. The network is based on a flexible and expandable infrastructure allowing easy node management. A support for the 6LoWPAN standard makes nodes reachable from outside the network using IPv6 and provides an infrastructure to realize IoT applications.**

I. Introduction

The gravitational core formed by the concepts of Web 2.0 and the Internet of Things (IoT) [1] is shifting Web applications and services concepts towards wider integration and accessibility, in light of an anytime, anywhere, anything internetworking paradigm. The future Internet builds on these bricks to make up a dynamic entity, yielding novel means of interaction with services, other users, and the environment. Wireless Sensor Networks (WSNs) have been recognized as a very important block of this internetworking concept. Tiny, distributed objects as they are, WSNs constitute a reasonably cheap sensory extension to Internet-connected devices; moreover, their computational capabilities allow for further (though possibly limited) use flexibility and functional expansion. Any kind of next-generation Internet-enabled portable device will set up advanced interactions with the "things" making up the new IoT, resulting in a pervasive infrastructure of fixed and mobile heterogeneous nodes, seamlessly providing, exploiting or sharing context-based services and applications. In particular, capturing the context and surroundings of devices will constitute a key component, making such operations as "Googling" physical reality possible and common [2]. Such a wide perspective requires stable foundations, starting from a widely agreed upon protocol and communication infrastructure, which has currently been identified in the IPv6/6LoWPAN protocol suite [3]. By integrating any object into the IP infrastructure, 6LoWPAN is an important enabling technology allowing to merge newer and older Web services, as well as to support the cited IoT interaction paradigm, while still running everything over the widespread Internet infrastructure.

As part of the SENSEI [4] consortium and in the context of the WISE-WAI project [5], the University of Padova is putting this vision into practice, by channeling experience in the field of wireless sensor networking towards the realization of a scalable and easily extendable network structure, which is basically formed of three classes of nodes (base stations, mobile nodes and specialized nodes) running compatible code but providing different functions and carrying out different tasks. The network spans the floors of three buildings, and includes high- as well as low-density areas. Nevertheless, our focus is not only on the setup and installation of the network (although this is by itself interesting and challenging), but rather on its use, development and structure optimization. To this end, all nodes are IPv6-compatible, which makes them directly addressable from any Internet-capable device. The nodes support diverse services, from environmental parameter monitoring to localization, and are in turn supported by lower-level functionalities allowing, e.g., to switch the application being run on the node, to change the class/role of a node, to spread software changes and updates over the air, or to perform low-level resets in case of malfunctions [6]. Offering services through our network provides an opportunity to realize part of the IoT vision and continue research efforts in the field: in addition, it demonstrates the advantages of the IoT in the management as well as everyday occurrences of University life, as explained in Sec. III.

II. Related work

The interconnection of WSNs to the Internet has been widely researched in the last few years. At the beginning of the WSN era, researchers focused on the development of dedicated systems, where highly specialized but non-standard protocols were used within the WSN, whereas one or multiple gateways were used to translate messages and ultimately connect the WSN to the external IP world. While these systems were generally efficient in the specific application scenario they were designed for, they lacked flexibility: developing new applications on top of them was therefore time-consuming and cumbersome, as it required modifications to the specialized protocols within the WSN. As a remedy to that, the 6LoWPAN standard has recently been proposed as a viable method to bring IPv6 to WSNs [7], [8] so that sensor nodes can be natively addressed and connected through the IP protocol. This has obvious advantages such as rapid connectivity and compatibility with pre-existing architectures, plug-and-play installation of WSNs, rapid development of applications as well as the possibility of integrating with existing Web services developed for standard IP networks.

Web services are extensively and successfully used mechanisms in Information Technology (IT) systems; they may be defined as techniques to develop interoperable and distributed applications exploiting Web standards like HTTP. As discussed in [9], sensor networks can greatly benefit from their usage, as Web services allow the integration of WSNs into any system that is built on standard IT components, e.g.,

978-1-4244-5328-3/10/$26.00 ©2010 IEEE

industry/home automation as well as home energy management systems. TinyREST [9] efficiently implements Web services on WSNs by carefully minimizing the overhead introduced by the transport layer whilst using data formats such as XML and WSDL. Web services can be classified into SOAP-based and RESTful-based. The authors of [10] have recently demonstrated that SOAP-based Web services are doable for WSNs. The RESTful approach is however currently preferred for these networks due to its lightweight character [11]. RESTful has the sensor resource as its main abstraction and every resource in every sensor node is linked and retrieved through a Uniform Resource Locator (URL). In addition, standard HTTP methods such as GET, POST, PUT and DELETE are used respectively to acquire, modify or delete the value of a given resource.

Recent research efforts explore the feasibility and the performance limits of the RESTful approach for Web services on top of 6LoWPAN for WSNs. These studies aim at improving the usability of WSNs, making them suitable for complex installations, while retaining the flexibility of IP-based networks. A recent paper [11] presents an IP-based WSN where nodes send data using Web services. The authors of this paper show that this approach is doable for resource constrained sensor nodes in terms of acquisition time for the sensor data and power consumption. In addition, they prove that TCP can be supported under particular network settings. A similar approach has been presented in [12] where WSN resources are integrated into IP-based networks exploiting Web services.

Along these lines, in this paper we present our implementation of a Binary Web Service (BWS) [13] for WSNs. Resources are handled according to the RESTful approach and binary encoded XML is used to reduce the transmission overhead. In addition, we exploit standard interfaces for access (Resource Access Interface, RAI) and publication (Resource Publication Interface, RPI) of resources. The peculiarity of our work is that of presenting an actual system, using standard protocols to offer various WSN services to both the administrative staff and regular users of the University (i.e., students and professors). We present our current state of the art and what is planned in the future to realize a complex WSN system; the resulting architecture will be able to provide network services to a specific, custom-designed base-station, as well as to generic mobile nodes accessing the network using standard protocols.

III. Scenario

A University offers many application scenarios for demonstrating the advantages brought about by the IoT in real life. In particular, at least three service categories can be offered in a University facility: i) Office automation: the services belonging to this category are automated applications, meant to help, e.g., the management staff; ii) Teaching: this category includes all functionalities that can be exploited by full registered users only; iii) Guest: visitors can access this class of services to retrieve basic information (e.g., to navigate around buildings) or to locate people.

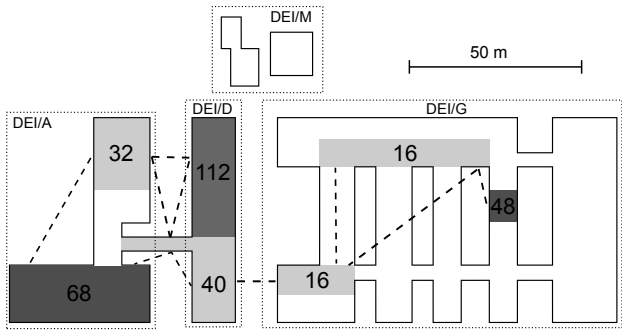

Figure 1. Graphic representation of the testbed at the University of Padova: the installation has been completed in all dark grey sections, and is being carried out in light grey sections. Dashed lines highlight wireless connectivity among far sections.

These services involve quite diverse technology, security and quality requirements. For instance, the first category includes such services as door access control, which aims at granting access to qualified persons and therefore needs integration with identification technologies (e.g., RFID) and a reliable backbone network, Ethernet in our case. Moreover, this application calls for complex security features, since the network needs both to authenticate the users entering a certain area and to avoid granting passage to forged IDs. On the contrary, guest services do not need a particularly high security level, but must be able to manage a large number of users, hence requiring high scalability. In fact, in order to route visitors through the building, the system needs to run a reliable positioning system while, at the same time, reporting information (such as location) to the users. This can be performed with low-power radio interfaces, such as the IEEE 802.15.4, so that users only require a USB dongle for setting up communications with the fixed network. Even though our testbed [6] can offer each of these services, in this paper we only focus on those enabled by the wireless sensor network backbone. In its actual configuration, outlined in Figure 1, our testbed consists of more than 200 static sensor nodes (currently being expanded to 300) and 100 more nodes that can be used both as mobile stand-alone devices or as USB devices for laptop connection.

By using Synapse++ [14], a fast and reliable over-the-air reprogramming system, the static backbone can be programmed with any of the following applications: i) Web Services; ii) Localization; iii) Experimental protocols. Experimental protocols and Localization can be installed on demand in an arbitrarily large fraction of the network; however, Web Services [13] constitute the default application being run by the nodes using the Binary Web Service (BWS) protocol.

In more detail, the BWS communicates through the IEEE 802.15.4 radio using the IPv6/UDP [3] protocol stack, thus enabling interoperability between our testbed and the Internet: in other words, it will be possible from any browser to open a page on the IP address of a specific node and look up/subscribe to its offered services, or read its sensor data. The majority of our nodes offer baseline sensing capabilities (light, temperature and humidity) as well as

some management parameters (battery level, transmission power); however, nodes installed in specific locations are programmed with additional services: for instance, sensors in proximity of the teaching rooms can be asked for the schedule of that room as well as a booking service; offices have sensors that can record the name of the visitor and the time of the visit; furthermore, the administrative staff can use the testbed in order to read the temperature in the whole building, and steer it to a comfortable level for employees and students.

We are currently working on multiple aspects: technology, smart automation and communications; we will connect our testbed to available actuators, such as heaters, air conditioners, light switches, etc. and a number of new devices, such as an automated door key lock with RFID, routing monitors for visitors and so on. In terms of smart automation, we are developing control mechanisms to monitor the environmental status of the buildings and generate reports if critical or abnormal conditions are found. The communication aspect is of fundamental importance, as we aim at connecting every device using standard mobile protocols either by browsing through gateway nodes or by directly accessing nodes through native BWS. Thanks to the aforementioned possibilities, our wireless sensor network becomes much more than an experimental research tool, turning it into a complete infrastructure allowing University users to experience typical IoT services [1].

IV. Node characterization and paradigm required

Notwithstanding the constraints (especially in terms of nodes computational power and storage capabilities) of the aforementioned scenarios, support to standard protocols adapted for operation in a WSN (such as 6LoWPAN/UDP) is highly recommended in order to achieve interoperability and integration with current Internet-aware devices. In light of these considerations, a minimal set of protocols encompassing all required functionalities is to be selected, in order to minimize complexity by maximizing code reuse. To keep network operations efficient, the architecture of the network hosting these functionalities should be scalable and easily extendable. To this end, we envision a resource-oriented paradigm, whereby heterogeneous services are provided both to sinks and to mobile nodes, which may be heterogeneous and not designed to receive a custom service in a specific network.

We distinguish among three types of nodes with correspondingly different feature sets, depending on node mobility, operating range, and level of specialization: i) Base Station Node (BSN), e.g., an IPv6 sink/router; ii) Mobile Node (MN) (e.g., wireless dongle to add WSN connectivity to a standard laptop); iii) Specialized Node (SN) (e.g., offering services like temperature readings or actuation). BSNs are usually static nodes, bearing no specialization and a network-wide operating range; to this end, they must be provided with bidirectional and simultaneous communication with one, many and possibly all nodes in the network, also exploiting data aggregation techniques as appropriate. Usually BSNs have direct connection to the Internet and can provide connectivity to the WSN; a dedicated channel for receiving events and subscribed data is also required. MNs, in a typical use case, are external nodes running no specific firmware for the WSN in use, but rather featuring compatibility with the network specifications and protocols. Given such compatibility MNs should be provided zero-knowledge access to any particular node in the network, possibly including BSNs; to this end, network probing, direct access to resources, temporary network join and resource subscription are relevant features to be supported. Finally, SNs are nodes in charge of delivering one or more very specific services, which makes SNs become a core part of the network, and potentially the most limited devices. Even though they are specialized, they might be in charge of diverse activities: they need to serve requests by BSNs, MNs or even other SNs requiring cooperation or relaying.

This preliminary description allows us to identify a set of requirements that should be supported by the network communication paradigm. In light of the interoperability and integration of the network with Internet-based communications, we choose to employ the Representational State Transfer (REST) paradigm [15] well known in the Internet domain, whereby any resource is addressed by a unique identifier of standard format. The features to be supported are summarized as follows: i) direct simple resource request/response; ii) concise one-to-many resource request/response; iii) structured resource request/response; iv) resource subscription and event or delayed notification; v) zero-knowledge network probing.

While being powerful enough to address interoperability, REST makes the access to any resource as easy as a web server interrogation. REST also simplifies the development of the network communication paradigm, which can be built around a single protocol by properly leveraging our flexible Binary Web Service [13] implementation in a versatile resource-oriented node design.

V. TinyOS Node Implementation

A. Binary Web/XML Services

The Binary Web Service (BWS) protocol [13] is a binary, scaled-down realization of REST, but compatible with the verbose HTTP protocol [16]. BWS is based on UDP, enabling the REST interaction model on severely limited devices such as wireless sensors. To access a resource through BWS, a request message is issued. In the 2-3 bytes long header of the message are specified the target resource (identified by a URL), the access method (GET, PUT, POST, etc.) and the format of the payload (Content-Type); the payload contains any data required to fully describe the request. A response describing the result of the request is sent by the receiving resource to the requesting entity, which is identified by its source (IP, port) pair; the header of the response contains the HTTP status code summarizing the result and the Content-Type (if any). All the aforementioned fields are encoded in binary form in a simple, short header; full support for URL strings is preserved but, alternatively, the target resource can be specified using a binary code.

BWS delegates payload data compression, and advocates the use of Efficient XML Interchange (EXI) from W3C [17] for Binary XML encoding. This choice is mainly due to the chance of operating the encoder in a schema-informed byte-aligned mode [17] which makes coding simpler and reduces the output size. However, we note that building the grammar and thus the specific implementation from a schema is quite complex; moreover building any generic EXI encoder is indeed difficult, but simple implementations can be derived only by analyzing specific schemas and by deriving the subset of features required by those schemas.

B. Implementation description

In our implementation, the communication with any resource is uniformed using common, flexible components (e.g., Binary XML Services), through a single-instance and resource-shared BWS implementation on top of the UDP/6LoWPAN stack. The same BWS module is designed to provide both server and client capabilities. It supports opening multiple servers, each able to serve parallel incoming requests from clients; as a client, it can handle concurrent communication with multiple different servers; therefore our implementation allows flexible services to be based on a single, reusable component. Both the client and the server entities provide support for URL requests or compressed URL requests, i.e., numerical identifiers (ID).

The BWS component provides server functionalities through the `BWSServer` interface, triggering an event for each incoming request to the registered resource.

```
interface BWSServer{
  event error_t request(
    uint16_t rid,
    uint8_t id,
    uint8_t method,
    uint8_t content_type,
    uint8_t *content,
    uint16_t length );

  event error_t requestURL(
    uint16_t rid,
    uint8_t *url,
    uint8_t method,
    uint8_t content_type,
    uint8_t *content,
    uint16_t length );

  command void response(
    uint16_t rid,
    error_t status,
    uint8_t content_type,
    uint8_t *content,
    uint8_t length);   }
```

The interpretation of the method and content is left to the appropriate resource depending on the requested web service, which is identified by the server port and the URL or ID. Every request is identified by a 16 bits locally unique field (`rid`), in order to support triggering multiple requests to the same component. A web service can still be configured to handle one request at a time by refusing further inquiries using an appropriate status response; more advanced components can track multiple requests and independently respond to each. The BWS module keeps track of every active request, by mapping every `rid` to the requesting node IPv6 address and UDP port. Responses can be sent asynchronously issuing a `response` command to the BWS server and providing the `rid` matching the request being served.

Remote BWS servers can be accessed through a complemental interface (`BWSClient`), which provides methods to access BWS features. On a client, every request is mapped to a command which requires a service to the BWS component; the corresponding response is an event triggered by the BWS module. To support clients sending concurrent requests, a 16-bit `rid` is associated to each request and returned to the resource as the result of the `request` or `requestURL` command. The response event is fired within the initiating component when the response message is received.

```
interface BWSClient{
  command uint16_t request(
    ip6_addr_t *addr,
    uint16_t port,
    uint8_t id,
    uint8_t method,
    uint8_t content_type,
    uint8_t *content,
    uint8_t length );

  command uint16_t requestURL(
    ip6_addr_t *addr,
    uint16_t port,
    uint8_t *url,
    uint8_t method,
    uint8_t content_type,
    uint8_t *content,
    uint8_t length );

  event void response(
    uint16_t rid,
    error_t status,
    uint8_t content,
    uint8_t *payload,
    uint8_t length);   }
```

The versatile design of the BWS component has been enabled by a proper modification of the `6lowpan` component interfaces. The `IP6P` component has been re-engineered to exploit the flexibility of parameterized interfaces as used in a similar manner by the `ActiveMessageC`; this allows the BWS module to take full control over the 6LoWPAN subsystem using the `UDP` interface described below. Note that the component `IP6P` parameterizes the interfaces depending on the local UDP port: for this reason, it does not appear among the function arguments. This modification allows easy code reuse for processing incoming packets or for writing outgoing packets to different UDP ports.

```
interface UDP {
  command error_t sendTo(
    const ip6_addr_t *addr,
    uint16_t port,
    const uint8_t *buf,
    uint16_t len );
  event void sendDone(
    error_t result,
    void* buf);
  event void receive(
    const ip6_addr_t *addr,
    uint16_t port,
    uint8_t *buf,
    uint16_t len );
```

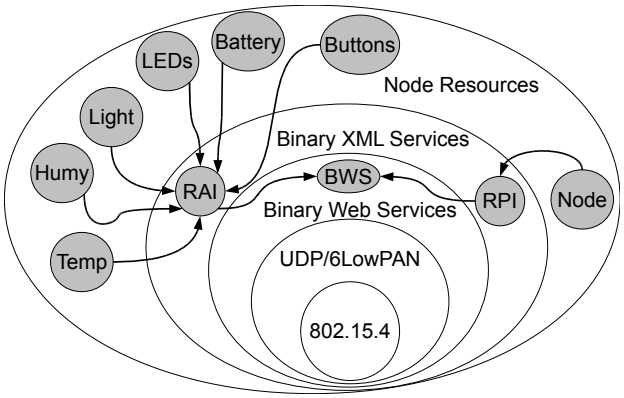

Figure 2. Components wiring in the SENSEI node.

VI. SENSEI CASE-STUDY

Our framework implementation is optimized, so as to allow easy provision and configuration of real-life resources. In the context of the European SENSEI project [4], the BWS module has been then connected to two different Binary XML Services, Resource Access and Resource Publish, providing project-specific node interfaces.

The Resource Access Interface (RAI) provides access to specific resources identified by the URL or the ID specified by the BWS module. BWS methods are mapped as follows: i) GET provides a reading of the current value of the resource; ii) PUT sets, if applicable, the resource value or inputs a new command; iii) POST is used to subscribe to the resource by setting an appropriate criterion to push notifications directly to the sender. The Resource Publish Interface (RPI) is used instead to provide a comprehensive description of node properties and offered Web services to a BSN.

RAI and RPI communication is based on an out-of-band agreement on an XML schema representing the information to be conveyed in the various operations, and then on an XML content exchange whenever required. EXI [17] has been selected as the standard format for Binary XML: however a full implementation of an EXI encoder/decoder is not advisable for a SN; therefore, a simplified implementation is to be preferred.

By investigating the EXI standard and the agreed XML schemas, the EXI coded schemas have been mapped to static and variable parts; furthermore, the process of encoding and decoding an EXI request body is done as follows: for every resource, headers, footers and separators are known; between such tags, resource-related values are read or written according to the simple algorithm required to encode/decode numeric values in the EXI coding.

Considering the well-known `telosb` sensor node architecture [18], we evaluated the ROM/RAM usage of the system described above, and summarized it in Table I. We highlight that implementation of the Binary Web Service module makes efficient use of both ROM and RAM size, independently of the number of clients or servers required by the upper layers. The RAI component translates request and associates them to the appropriate resource; it also implements a highly specialized EXI encoder/decoder which proves to be effective in terms of ROM occupancy. Resource components (temperature, humidity, light, etc.) require a larger memory footprint mainly because a specialized driver component is required for every on-board chip. RAM occupancy, on the other hand is low with respect to the available space (approximately 10 kB), except for the UDP/6LoWPAN implementation which requires a large static RAM allocation, mainly due to a 1280-byte buffer required for re-assembling fragmented datagrams.

Table I
TINYOS COMPONENTS ROM/RAM UTILIZATION

Component	ROM	RAM
TinyOS core	1398	4
802.15.4 and ActiveMessage	9418	328
UDP/6LoWPAN	5182	1936
BWS	2950	326
RAI/RPI	1374	156
Resources	9800	354

VII. LESSONS LEARNED, VISION AND FUTURE WORK

Thanks to the experience acquired up to the current stage of the SENSEI project, we have envisioned a next-generation system which leverages on the strengths of the architecture set up to date. A Web Service model for WSNs has proved to be valid for the current purposes and a careful implementation is strictly required to adapt this communication model to a limited sensor node. The versatility of our implementation, together with the flexibility of Web services allows to make further steps towards a fully integrated system built around the BWS component.

Our vision now requires that every interaction (e.g., as devised at the end of Section IV) is managed internally by the BWS. Each interaction will be mapped to use standard REST methods, paired with a proper XML definition of the data required in the process, to allow strong code reuse even for very different operations or services offered.

As shown in the previous section direct simple request/response interactions have been easily implemented, even though a specific interaction to gather many responses from different nodes through a single request (concise one-to-many communication) has been currently left as a future work. Another interesting feature required by next-generation systems is support for structured requests, required to gather multiple values from the same node; also, interpreting complex requests based on the current state of the resources (e.g., turn on the air conditioning system in rooms with temperature higher than a certain threshold and where the lights are turned on) is also a required function that can be provided. In any event, the previous interactions will be implemented thanks to the BWS component flexibility which, together with the versatility of Web services, can support complex XML interactions without redesigning the paradigm and the components already in use.

In this vision EXI coding plays a central role, as binary XML coding should be easy to implement and should allow strong code reuse in order to facilitate the implementation

of multifaceted Web services on sensor nodes. However, our understanding of EXI format has led to the conclusion that the procedure required for coding two different XML schemas with minimal differences could be completely different, so a minimal variation of the schema may require a very different implementation. In this light, we have started evaluating the EXI coding for sensor nodes, by building an XML schema pre-processor that will directly output the variable part of the C code required to encode/decode that schema; as a second step, we will supply the pre-processor with a set of optimizations aimed at minimizing the output code size.

The last step in building our next-generation network will be the standardization of the offered resources and services. This will be accomplished by assigning a URL to all resources in a standard, reusable and extendable fashion, and then by mapping the procedure to opt-in and configure guest mobile nodes to a web service URL. We are confident that such a system can be easily replicated in different scenarios through small changes specific to the different resources and services offered, but without requiring any modification to its core architecture.

In order to realize the Internet of Things, the simple, efficient way of realizing the REST architecture presented in this paper is surely needed as a global standard. Although TCP/HTTP/XML is useful for some applications and more powerful networks and devices, it is inappropriate for a huge range of uses. Recently a new standardization effort has been started at the IETF called 6lowapp [19] with the goal of realizing application layer paradigms for constrained networks and devices.

VIII. Conclusions

This paper presented an integrated framework for interconnecting WSNs and actuators to standard networks as Web services. This is achieved through shared standard interfaces working with scalable lightweight protocols, such as 6LoWPAN. The framework has been validated through a case study realized under the guidelines of SENSEI, one of the largest European project on WSNs, and through the actual implementation of the services developed on the department-wide WSN testbed set up in the Information Engineering Department (DEI) at the University of Padova.

Even though some of the described functionalities are still under realization, our WSN installation can already offer many different services ranging from localization to data gathering and actuation control. The next steps in the development of our infrastructure will deal with seamless interoperability with available services and with the further extension of the catalogue of resources available at the University.

Acknowledgment

This work has been supported in part by the Cassa di Risparmio di Padova e Rovigo Foundation, Italy, under the project WISE-WAI, http://cariparo.dei.unipd.it, and by the FP7 EU project "SENSEI, Integrating the Physical with the Digital World of the Network of the Future," Grant Agreement Number 215923, http://www.ict-sensei.org.

References

[1] Future Internet Assembly, "European Future Internet Portal." [Online]. Available: http://www.future-internet.eu/

[2] "Sense & Sensitivity by Orange Lab." [Online]. Available: http://senseandsensitivity.rd.francetelecom.com/index.php

[3] Z. Shelby and C. Borman, *6LoWPAN: The Wireless Embedded Internet*. Wiley, Nov. 2009.

[4] EU Integrated Project, "SENSEI: Integrating the physical with the digital world of the network of the future." [Online]. Available: http://www.ict-sensei.org/

[5] "WISE-WAI project web site." [Online]. Available: http://cariparo.dei.unipd.it

[6] P. Casari *et al.*, "The WIreless SEnsor networks for city-Wide Ambient Intelligence (WISE-WAI) project," *MDPI Journal of Sensors*, vol. 9, no. 6, pp. 4056–4082, Jun. 2009. [Online]. Available: http://www.mdpi.com/1424-8220/9/6/4056

[7] A. Dunkels and J. P. Vasseur, "IP for Smart Objects," IPSO Alliance White Paper No. 1, Sept. 2008.

[8] J. W. Hui and D. E. Culler, "IP is Dead, Long Live IP for Wireless Sensor Networks," in *Proc. of ACM SenSys*, Nov. 2008.

[9] T. Luckenbach, P. Gober, S. Arbanowski, A. Kotsopoulos, and K. Kim, "TinyREST - a protocol for integrating sensor networks into the internet," in *Proceedings of REALWSN*, Stockholm, Sweden, Jun. 2005.

[10] B. Priyantha, A. Kansal, M. Goraczko, and F. Zhao, "Tiny web services: design and implementation of interoperable and evolvable sensor networks," in *Proceedings of ACM SenSys*, Raleigh, NC, Nov. 2008.

[11] D. Yazar and A. Dunkels, "Efficient Application Integration in IP-Based Sensor Networks for Emerging Energy Management Systems," in *Proceedings of ACM Buildsys*, Berkeley, CA, US, Nov. 3 2009.

[12] L. Schor, P. Sommer, and R. Wattenhofer, "Towards a Zero-Configuration Wireless Sensor Network Architecture for Smart Buildings," in *Proceedings of ACM Buildsys*, Berkeley, CA, US, Nov. 3 2009.

[13] Z. Shelby, M. I. Ashraf, M. Luimula, J. Yli-Hemminki, and A. P. Castellani, "BinaryWS: Enabling the Embedded Web," Coimbra, Portugal, submitted to EWSN.

[14] M. Rossi, N. Bui, G. Zanca, L. Stabellini, R. Crepaldi, and M. Zorzi, "Code Dissemination in Wireless Sensor Networks using Fountain Codes," *IEEE Trans. Mobile Comput.*, 2010, accepted for publication.

[15] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, Irvine, 2000. [Online]. Available: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

[16] R. T. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," IETF RFC 2616, 1999. [Online]. Available: http://www.ietf.org/rfc/rfc2616.txt

[17] J. Schneider and T. Kamiya, "Efficient XML Interchange (EXI) Format 1.0," W3C Working Draft, 2008. [Online]. Available: http://www.w3.org/TR/2008/WD-exi-20080919

[18] CrossBow, "TelosB Mote Platform." [Online]. Available: http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf

[19] "IETF 6LowApp wiki." [Online]. Available: http://6lowapp.net