

LoWPANs Meet Service-Oriented-Architecture

Mohsen Rouached*, Shafique Chaudhry*, Anis Koubâa*§

*COINS Research Group, Al-Imam Mohamed bin Saud University (CCIS-IMAMU), Riyadh, Saudi Arabia.

§CISTER Research Unit, Polytechnic Institute of Porto (ISEP/IPP), Portugal.

Emails: Emails: {rouached,shafique,akoubaa}@coins.csrlab.org

Abstract

The Low-Power Wireless Personal Area Networks (LoWPANs) have been recognized as a promising technology for ubiquitous and pervasive computing systems. However, LoWPAN technology is still open to being adapted to existing interoperability mechanisms defined for the Internet. Service-Oriented-Architecture (SOA) is one of the key paradigms that enables the deployment of services at large-scale over the Internet domain and its integration with LoWPANs has opened new pathways for novel applications and research. Despite the need to integrate SOA with LoWPANs, only a handful efforts are underway to achieve the goal. In this paper, we discuss the integration of LoWPANs with Service-Oriented-Architecture (SOA) for seamless provisioning of services in LoWPANs, especially considering the future of LoWPANs, i.e., IPv6-enabled LoWPANs (6LoWPANs). We first present an overview about the general concepts of SOA and its applicability onto low-power devices. Then, we discuss 6LoWPAN, a milestone protocol that bridged the gap between low-power devices and the IP world and discuss the advantages, challenges and opportunities for porting SOA over LoWPANs. We also present the main research efforts that contributed to featuring the integration LoWPANs with SOA and we draw a research roadmap on potential research directions and challenges for achieving an efficient coupling among LoWPANs and SOA.

I. INTRODUCTION

Cyber-Physical Systems (CPS), Cooperating Objects and Internet-of-Things (IoT) represent, these days, the current trends of computing philosophy. This new paradigm goes beyond the traditional vision of considering a network as a digital data sharing infrastructure decoupled from its environment to considering a network as a physical objects/events sharing infrastructure where data is no longer decoupled from its physical environment. This new vision gave rise to the concept of ubiquitous and pervasive monitoring (and control) applications used in different areas such as smart home automation, industrial monitoring, healthcare, intelligent transportation systems, surveillance systems, etc. Wireless Sensor Networks and RFID systems are the key enabling technologies for these pervasive systems, as they provide an interface between the physical world (environment) and the digital and cyber worlds (computing devices and the Internet).

While these technologies have been evolving rather independently, there is currently increasing trends towards integrating them all together. The Internet, always known as the core backbone for large-scale distributed systems, is considered the main prospective mediator to interconnect all these heterogeneous systems. Though LoWPANs have been considered as an essential technology for pervasive environments, the recently emerging IPv6 over Low-Power Wireless Personal Area Networks, (6LoWPAN) standard has given rise to the concept of Wireless Embedded Internet [1]. It gave rise to a new paragon that bridged the gap between low-power wireless networks and the IP world. 6LoWPANs present the key advantage of enabling end-users to remotely and seamlessly access low power embedded devices, such as wireless sensor nodes,

through the Internet using IPv6 as the underlying network protocol. This new paradigm opens the door to several new challenges, for adapting/adopting legacy Internet solution in 6LoWPAN, which are inherited from the resource-constrained nature of low-power sensing devices. Arguably, 6LoWPAN considers the fulfillment of two main requirements namely *energy-efficiency* and *interoperability*. While the energy issue has been carefully and extensively addressed by several proprietary solutions, interoperability has been ignored to some extent in earlier design objectives [2].

Interoperability requirement has been addressed through the adaptation of Internet mechanisms to the requirements of WSNs. While 6LoWPAN achieve interoperability at Network Layer, Service-Oriented Architecture (SOA) has been considered for the same purpose at the application layer. In fact, Service-Oriented Architecture (SOA) is one of the core mechanism for service deployment in the Internet that has been adapted in WSNs for making easier and more effective service deployment . It possesses an architectural style encompassing a set of services for building complex systems from existing components. As an architectural evolution and a paradigm shift in systems integration, SOA enables the discovery, access and sharing of the services, data, computational and communication resources in the network for multiple users. It also allows rapid and cost-effective composition of interoperable and scalable systems based on reusable services exposed by these systems. SOA inherently supports two major requirements: heterogeneous infrastructures and run-time adaptability, which are essential for large-scale cyber-physical systems in which multiple applications run over diverse platforms and adopt different technologies.

Though SOA has become a cornerstone in many recent

research efforts, many of its elegant potentials have not been sufficiently explored in LoWPANs. In this paper, we tackle integration of SOA with LoWPANs and discuss the underlying challenges and benefits. We especially focus 6LoWPAN technology because it is essential the future of LoWPANs, envisioned to create an Internet of Things. Our contributions in this paper are to discuss main research efforts that contributed to the integration of LoWPANs with SOA, and drawing a research roadmap on potential research directions and challenges for achieving an efficient coupling among (IPv6-enabled) LoWPANs and SOA.

The remainder of this paper is structured as follows. Section 2 presents the background materials about Service-Oriented Architecture and the 6LoWPAN protocol. In Section 3, we present the main research challenges towards the integration between SOA and LoWPANs with a case study of SOA requirements for 6LoWPAN. Section 4 describes a literature review about the current efforts of related works that proposed solutions to the SOA/LoWPANs coupling to illustrate how the raised challenges have been tackled. Finally, Section 5 concludes the paper and presents research roadmap and promising research areas that pertains to SOA and LoWPAN integration.

II. BACKGROUND

A. Service Oriented Architecture

The SOA paradigm defines a software architecture that comprises loosely coupled distributed components cooperating through a communication conduit, which enables the construction of composite services. SOA aims to bring about component reuse, irrespective of implementation language or host platform, and as such it can be thought of as simply an extrapolation of good software engineering practices, taking us from the *class reuse* concept to *service reuse* concept. Thus, SOA typically encompasses the following features:

- *Component architecture*: SOA is based on reusable software components enabling to build scalable heterogeneous (i.e. platform- and language-independent) service architecture.
- *Loose coupling*: The principle of Service Loose Coupling promotes the independent design and evolution of a service's logic and implementation while still guaranteeing baseline interoperability with consumers that have come to rely on the service's capabilities.
- *Platform independence*: This feature has been achieved by the adoption of standards, which have been the key mechanism enabling previously incompatible technologies to work cooperatively across a wide range of different platforms. Single services can interoperate with each other without depending on specific platforms or programming languages.
- *Transparency*: It is ensured by decoupling service functionalities from their actual implementation.
- *Flexibility*: SOA must ensure flexibility so as a system would be able to deal with dynamic changes of its configuration and behaviour according to varying requirements.

It results that the use of SOA enables to share, integrate and cooperate heterogeneous hardware and software components; Thus, distributed applications can be achieved with lower cost, better overall system utilization and performance. This makes it easier and possible for users to seamlessly access shared data, resources and functionalities that are not locally available.

1) *SOA: principles*: The main elementary concept in SOA is the *service*. A service is the mechanism through which entities that offer capabilities (service providers) and entities with specific needs (service requester/consumer) can interact. The interaction with services is regulated by a set of basic mechanisms that allow to offer, discover, interact with and use a service. A service is accessed by means of a service interface, which comprises the specifications of how to access the underlying capabilities.

SOA uses the Find-Bind-Execute paradigm as shown in Figure 1.

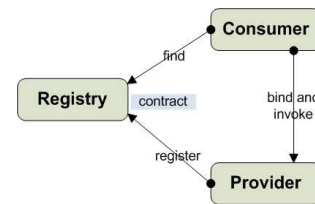


Fig. 1. The Find-Bind-Execute Paradigm

In this paradigm, service providers register their service in a public registry. This registry is used by consumers to find services that match certain criteria. If the registry has such a service, it provides the consumer with a contract and an endpoint address for that service. In a typical scenario, a service provider hosts a network-accessible software module, which represents an implementation of a given service, and provides a service description through which a service is published and made discoverable. A client discovers a service and retrieves the service description that will be used to bind to the provider and invoke the service.

For transparency purposes, a service is opaque in the sense that its implementation is typically hidden from the service consumer except for (1) the information and behavior models exposed through the service interface and (2) the information required by service consumers to determine whether or not a given service is appropriate for their needs.

2) *Web services*: Web Services (WSs) have been emerging as the leading implementation of SOA upon the Web. WSs have added a new level of functionality for service description, publication, discovery, composition and coordination extending the role of the Web from a support of information interaction to a middleware for application integration.

The basic Web Service protocol stack typically comprises four protocols:

- *Service Transport Protocol*: it is responsible for transporting messages between network applications. It includes

basic application-layer protocols such as HTTP, SMTP, FTP, as well as the more recent Blocks Extensible Exchange Protocol (BEEP).

- *XML Messaging Protocol*: it is responsible for encoding messages in a common XML format so that they can be understood at either end of a network connection. Currently, the commonly used protocols are XMA Remote Procedure Call (XML-RPC), WS-Addressing, and Simple Object Access Protocol (SOAP).
- *Service Description Protocol*: it is used to describe the public interface to a specific Web Service. The Web Services Description Language (WSDL) interface format is typically used for this purpose.
- *Service Discovery Protocol*: it centralizes services into a common registry such that network Web Services can publish their locations and descriptions, and makes it easy to discover what services are available on the network. Universal Description Discovery and Integration (UDDI)¹ protocol was specified for this purpose.

3) *REST technology*: REpresentational State Transfer (REST) [3, 4] was originally introduced as an architectural style for building large-scale distributed hypermedia systems. This architectural style is rather an abstract entity, whose principles have been used to explain the excellent scalability of the HTTP 1.0 protocol and have also constrained the design of its following version, HTTP 1.1. Thus, the term REST is very often used in conjunction with HTTP. The REST architectural style is based on four principles:

- 1) Resource identification through the Uniform Resource Identifier (URI). A RESTful Web service exposes a set of resources which identify the targets of the interaction with its clients. Resources are identified by URIs, which provide a global addressing space for resource and service discovery.
- 2) Uniform interface. Resources are manipulated using a fixed set of four create, read, update, delete operations: PUT, GET, POST, and DELETE. PUT creates a new resource, which can be then deleted using DELETE. GET retrieves the current state of a resource in some representation. POST transfers a new state onto a resource.
- 3) Self-descriptive messages. Resources are decoupled from their representation so that their content can be accessed in a variety of formats (e.g., HTML, XML, plain text, PDF, JPEG, etc.). Metadata about the resource is available and used, for example, to control caching, detect transmission errors, negotiate the appropriate representation format, and perform authentication or access control.
- 4) Stateful interactions through hyperlinks. Every interaction with a resource is stateless, i.e., request messages are self-contained. Stateful interactions are based on the concept of explicit state transfer. Several techniques exist to exchange state, e.g., URI rewriting, cookies, and

hidden form fields. State can be embedded in response messages to point to valid future states of the interaction.

In [4], authors used architectural principles and decisions as a comparison method to illustrate the conceptual and technological differences between RESTful Web services and WSDL/SOAP based Web services. Authors concluded that: On the principle level, both two approaches have similar quantitative characteristics. On the conceptual level, less architectural decisions must be made when deciding for WS-Web services, but more alternatives are available. On the technology level, the same number of decisions must be made, but less alternatives have to be considered when building RESTful Web services. For more details, the reader is referred to In [4].

B. 6LoWPAN: The future of LoWPANs

Tailored for low data rate applications on cheap devices, IEEE 802.15.4 standard redefines communication paradigm from networking (in literal sense) to connectivity in metaphoric sense. In this regard, IEEE-steered initiative of IEEE 802.15.4 standard befits really well as a candidate for ubiquity. These devices are poised to offer a broad range of services, such as smart homes, ubiquitous office environments etc. However, in order to exploit the full potential of ubiquitous features of these devices, we must connect them to the Internet.

In fact, the motivation for IP connectivity for LoWPANs is manifold:

- The pervasive nature of IP networks allows for the use of existing infrastructure and information resources, i.e. the Internet.
- IP provides extensive interoperability for other networks, e.g., sensor networks, and devices on other IP network links, e.g., WiFi, Ethernet, GPRS, etc., which means that IP-based devices can be more easily connected to other IP networks, without the need for translation gateways.
- IP-based technologies, along with their diagnostics, management and commissioning tools and services, such as Network Management (SNMP) [5], Neighbor Discovery (ND), Duplicate Address Detection (DAD), Router Discovery and Stateless Address Auto-Configuration (SAA)[6], already exist and are proven to be working.
- There exist established security mechanisms for authentication, access control and firewall for IP. It is worth mentioning that network design and policy mechanisms determine the access control and not the technology.
- For IP, it exists established Application Level data models and services, such as for instance HTTP, HTML, SOAP, REST etc. Additionally proxies architectures for the higher level services are also available.
- IP supports end-to-end reliability as well as link reliability.
- IP provides most industrial standards support.

Though the integration of IP with IEEE 802.15.4 brings in generous convenience for the users, it also presents a plethora of challenges to 6LoWPAN designers and implementers. These challenges owe greatly to the fact that both

¹<http://www.uddi.org/>

networks and in-network devices are exorbitantly different. The most fundamental difference lies in their packet sizes at the link layer. For IEEE 802.15.4 a packet size of 127 octets is standardized which, after considering frame header and link layers security options, could leave only 81 octets for the upper layers. This is obviously far below the minimum IPv6 packet size of 1280 octets. An adaptation layer that fragments IP packets for an IEEE 802.15.4 network into the transmittable size, and reassembles the packets likewise is essential. On the device level, IEEE 802.15.4 devices are constrained in form factor, most of all in battery, when compared against the peer devices in IP domain [RFC assumptions, challenges].

Extending the ubiquitous theme of IEEE 802.15.4 to include wider accessibility is the slogan of the 6LoWPAN [7], IPv6 over Low Power Wireless Personal Area Network. Standardized by IETF [8], 6LoWPAN integrates IEEE 802.15.4 (LoWPAN) with IPv6 with a view to enhance the connectivity of IEEE 802.15.4 devices from mere locality to the entire globe through the use of IPv6. The basic protocol stack for 6LoWPAN is shown in Figure 2.

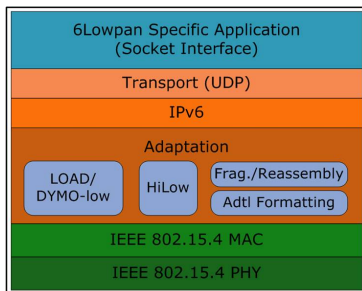


Fig. 2. The 6LoWPAN Protocol Stack

6LoWPAN's ability to sense-compute-communicate-and-(possibly) actuate by tiny devices has set-up a new wave of applications. Service providers, network operators, and brokers have all jumped to join the bandwagon of the new phenomenon that is very promising in terms of opening nice markets for them. They are convinced to introduce services ranging from home automation to telematics that are 'ubiquitous'. This means that these services must be cognizant of their environments and the devices around them, and must use them for value-addition. This ubiquitous paragon stems from the Mark Weiser vision two decades ago, allowing it to come to reality.

III. RESEARCH CHALLENGES FOR INTEGRATION OF SOA IN LOWPANs

A. SOA Challenges for LoWPANs

LoWPAN technology has established itself as a vital constituent for several future applications, however, this usage shall also present a number of challenges in terms of interoperability. There are a number of scenarios which can be presented as test cases for the need of interoperability, for example, a smart home with a set of services like security,

energy management, assisted living, etc. A home in this case would have intrusion sensors on doors and windows, smoke sensors in rooms, temperature and light sensors for temperature control and may be fire sensors connected to fire station. Traditionally, each sensor shall be running only one application restricting the generic extensibility of the infrastructure. If we could access all these sensors (and applications) through a common interface, not only we can continue to run the existing applications, but we can also create and run more applications using the same resources. The necessity, therefore, arises to espouse an interoperability architecture that is open and extensible, and allows for dynamic integration of services.

The enabling of an open and extensible architecture requires interoperability at network as well as at application level. Network layer interoperability can be best achieved using IP, therefore, 6LoWPAN is a strong candidate to achieve this objective. However, the existing protocol stacks for 6LoWPAN do not define policies for ensuring interoperability at service (or application) level. The application layer interoperability poses bigger challenges. Different types of sensors are available, which generate sensor-specific data. The application developer must understand and analyze the messages types and parameters used in the sensor nodes. One solution is to adopt a common specification (e.g. ZigBee, 6LoWPAN), for all the sensing devices. This approach may work for a small set of devices, but is highly impractical. An alternative approach is to tailor, trim and use existing standard services in a light-weight fashion. SOA is a promising candidate middleware platform that closes this interoperability gap and mediates data exchange between heterogenous sensor platforms and Web applications and services in a unified way.

The SOA, however, brings with it numerous research and development challenges for use on low power sensor nodes in general and for 6LoWPAN in particular. These challenges range from resource constraints of sensor nodes to the application space of such networks. In what follows, we present the most relevant challenges for integrating SOA in LoWPANs:

- *Resource constraints.* Size and energy consumption are the foremost constraints exhibited by LoWPAN sensing devices. Add highly limited bandwidth, processing power and memory resources and you get the exact picture of resource or lack of those. These restrictions allow on limited complexity message processing which makes the porting of traditional web services and SOA even more challenging.
- *Sensor Node's Duty Cycle.* The sensor nodes are generally battery powered and are expected to operate for a long duration with minimal duty cycles. It means nodes are awake for a certain periods while are 'asleep' for other intervals. This is in much contrast with typical web service hosts which are assumed to be always on.
- *Data driven services.* While traditional SOAs are based on a request-response message pattern, control applications running on embedded networks are typically data driven: data is acquired periodically at the sensors and pushed to connected services. These services produce new

data based on the received input which is consecutively pushed to the next service in the processing chain.

- *Data Life Cycle.* Service instances which abstract hardware devices, such as sensor services, may be used by multiple applications simultaneously. If one of these applications changes the state of the service, this is visible to other applications. Web service instances on the other hand are typically not shared and changes to one instance are not visible to other applications. SOAs for embedded networks therefore have to provide techniques to facilitate multi-user access.
- *XML.* Sensor nodes use a low powered radio which has a low data rate. The total amount of data sent cannot be very large to meet battery life and latency constraints whereas the payload of Web Service messages (i.e., SOAP messages) is virtually always XML. The grammar of these XML documents is specified using XML Schema. The advantages of XML are the vast tool support and its human-readable format. The downside is that it is very verbose resulting in large overhead in terms of size, which renders it unsuitable for the use in WSNs. This limitation also holds for HTTP as transport protocol for SOAP messages. The resulting length of network messages which are comprised of HTTP, TCP/IP and XML data easily surpasses the maximum packet length and bandwidth of radio interfaces available on typical sensor nodes. Apart from message length, also the amount of code required implementing HTTP, TCP/IP, and an XML parser exceeds the capabilities of sensor nodes by far.
- *Transport.* Usually, Web Service messages are conveyed using standard application-layer Internet protocols such as SMTP, FTP, or (virtually always) HTTP. Since these protocols reside on top of TCP/IP, they also inherently carry the overhead introduced by TCP/IP. To avoid the overhead of an application-layer protocol, there is also a transport-layer binding for TCP. The problem is that none of these protocols is applicable for resource-constrained devices like sensor nodes. Werner et al. [9], analyze these protocols in terms of their communication overhead. For a simple parameter- and return-less Web Service call, they report an overhead of 560 Bytes for HTTP, 576 Bytes for FTP, 2535 Bytes for SMTP, and 538 Bytes for TCP. The authors present PURE [9] as a lightweight Web Service transport protocol for resource constrained devices. PURE is based on UDP (User Datagram Protocol), but avoids its disadvantages by adding a message flow control and fragmentation feature. PURE itself has still a communication overhead of 66 byte. Since PURE does not provide an addressing scheme for Web Services, WS-Addressing² as transport protocol independent addressing mechanism has to be used causing additional communication overhead. The aforementioned values are strictly for this layer, e.g.,

for HTTP, TCP/IP overhead is not included.

B. SOA Requirements for 6LoWPANs

There exist a number of constrasting technical requirements that make the integration of 6LoWPAN with SOA extremely challenging. On one hand, 6LoWPANs are IPv6 networks; while at the same time, these are sensor networks that comprise a large number of nodes with extremely limited resources. Existing web services solutions for traditional IP networks cannot be applied directly because of the resource constraints. For example, a 6LoWPAN node may run out of energy causing a fault in the network. This node failure is a design feature of sensor networks as compared to other networks where it is less expected. As another feature, the applications, when designed for traditional networks may have restrictions in terms of performance and response time as compared to the hardware limitations, when designed for sensor networks. The traditional networks run a diversity of applications as compared to LoWPANs where the network is generally executing a single application in a cooperative fashion, though there are various proposals to run multiple applications at the sensor network. Furthermore, as an inherent WSN characteristic, 6LoWPAN could possibly be a data centric network which is different than traditional IP network behavior, however, on the contrary, because of IP support, there is a possibility that LoWPANs support a variety of services making it further complicated for network management operations. More interestingly, the sensor nodes may be deployed in a certain area and because of unpredictable situations, configuration errors or even environmental conditions can cause the loss of a partial or entire WSN even before it starts its operation. This situation is almost impossible for traditional IP networks operations.

In summary, the porting of SOA over 6LoWPAN could involve catering of the application level interoperability needs for the networks of hundreds or thousands of nodes which show enormous resource limitations yet providing IPv6 support. This situation demands for light-weight (in terms of both processing and communication) architecture with atleast the following characteristics:

- Be cognizant of the fact that it is to be deployed across its native network inside a wireless network. Such awareness at the traditional web services managers would entail changes (adaptability) to the communication across IPv6 and 6Lowpan networks.
- Take into consideration the fact that extra processing such as fragmentation/re-assembly may be carried out at the 6LoWPAN Gateway. For instance, the ingress query / response messages shall be parsed at the gateway and a corresponding query / message is generated inside the 6LoWPAN. Likewise, a response from within the 6LoWPAN terminates at the gateway and is encapsulated inside the respective link layer for the egress network.
- Be considerate of the compression of SOAP messages (including their XML payload) and the processing of the compressed data on 6LoWPAN nodes. Although

²Web Services Addressing 1.0. Tech. rep., 2006.<http://www.w3.org/2002/ws/addr/>.

approaches for the compression of XML exist in general, e.g. [10], none of them has been applied to SOAP messages in resource-constraint WSN environments.

- Considers the optimization of information flow and accessibility to allow all authorized applications to share this data.
- Be aware of the fact that novel architectures must be needed for application and data composition and incorporation of potential value-added decision making support.
- Be capable of providing an information and application web which can be accessed and extended, by authorized, users and applications, through standard interfaces.

IV. LITERATURE REVIEW

This section provides a brief description of current trends in employing SOA and Web Service-based technologies in WSNs and 6LoWPAN.

Several projects have been developed in order to provide an SOA approach for embedded networks, such as eSOA[11], SIRENA [12], SOCRADES [13], RUNES [14], and OASiS [15]. The majority of these projects aim at making embedded devices directly accessible with Web Service technologies by installing an adopted Web Service stack, i.e. the Devices Profile for Web Services (DPWS) stack [16]. However, while this approach is suitable for a certain range of devices, there will always be a class of very small and lightweight devices, which cannot deal with the additional overhead introduced by the Web Service technologies, and consequently, require more efficient SOA implementations.

A. DPWS-based approaches

The Devices Profile for Web Services (DPWS) [16] (see Figure 3) was developed to enable secure Web service capabilities on resource-constraint devices. It features secure exchange of messages, dynamic discovery and description of Web services, and subscribing/receiving events to and from a Web service. DPWS can also be used for inter-machine communication. However, the latter requires the devices to have an implemented peer functionality (i.e. a specific DPWS client implementation) to use a corresponding service hosted on another device.

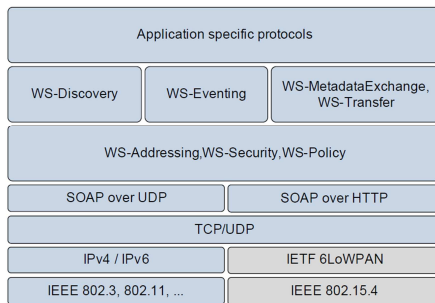


Fig. 3. Devices Profile for Web Services [17]

Some recent works, such as [18–20], have discussed the adequacy of DPWS for WSNs. These works recommended to eliminate the use of SOAP and HTTP protocols due to their high overheads. Instead, they provide some solutions based on application-specific-formats that are used in the proposed Tiny DPWS protocol stack. Although the proposed application-specific-formats reduces the size of the transmitted messages in the network, it hinders the extensibility of the solutions. For any new service to be offered by sensor nodes, a new application-specific-format should be defined in order to make it work in the proposed infrastructure.

After discussing a set of potential technologies that could be deployed to overcome the problems of heterogeneity and interoperability in WSN and evaluating their advantages and disadvantages, In [17] describes a SOA based middleware which mediates data exchange between heterogenous sensor platform and Web applications and services in a unified way. The idea of this middleware consists of introducing several modifications on the original version of DPWS as depicted in Figure 4.

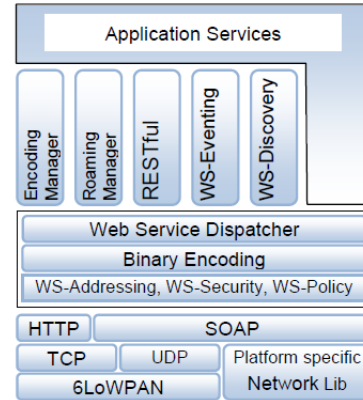


Fig. 4. DPWS-based Architecture [17]

The proposed architecture relies on some optimization techniques to reduce the overhead imposed by traditional Web Service technologies:

- Binary encoding techniques are used to overcome the problem of huge overhead of XML messages.
- The less powerful nodes can alternatively offer their functionality in the RESTful interface in order to exploit the benefits of using web services without suffering from their complexities.
- WS-eventing is used to save the limited network bandwidth. Indeed, instead of calling the desired service periodically, the user can simply subscribe to the eventing interface of this service. WS-eventing notifies the clients when the requested service/data has changed according to the request definition.
- The mapping between WSN and Internet applications is ensured by the address mapping module. This module employs stateless address mapping to facilitate communication between these networks.

- The Mobility Manager enables a node to leave one platform type and attach another one without interrupting the open service transaction.
- To accommodate heterogeneous sensor nodes, the proposed protocol stack works on top of 6LoWPAN as well as other platform specific networking libraries.

Although this approach appears very interesting and promising, implementations aspects and performance measures are not discussed. Moreover, interactions between the different components are not studied.

In [21], Moritz et al. presented different XML specific and XML non-specific compressors and their influence on message size of the Devices Profile for Web Services (DPWS). Therefore, a test scenario was analyzed with 18 different messages. They focused on the SOAP compression to makes DPWS applicable for deeply embedded devices in 6LoWPAN networks, which are characterized by very constrained resources such as small computing power, limited power supply, and a few tens of storage capacity. The results showed that most existing compressors suffer from the simplicity of XML structures, which are the results of non complex services deployed on the deeply embedded device. Only the Efficient XML Interchange (EXI) and Fast Infoset (FI) formats provide a much better compression rate, because of the usage of XML schema definitions to include further structure information. Usage of compression after re-encoding has a minor influence. Details about compression techniques and their performance results are available in [21].

To increase parsing performance, a new encoding Devices Profile for Web Services (encDPWS) approach was introduced in [22]. This paper investigated the applicability of DPWS in 6LoWPAN networks. Their main objective is to optimize the message encoding process in order to reduce the overhead of this SOAP-based protocol. The encDPWS encoding is based on the Tag-Length-Value (TLV) format, which is extensively used in lower layers (e.g. IPv6 extension header chaining) and is applicable on resource-constrained devices. Required buffers are allocated before starting the parsing process, and non-supported fields are left out without parsing. Carrying length information for every data unit inline differentiates encDPWS from the other well-known solutions, and significantly increases the parsing performance.

Based on some measurements and comparisons, Moritz et al. showed that it is possible to compress SOAP-based Web service down to fit the size requirements of low power deeply embedded devices. This allows DPWS deployments in 6LoWPAN networks, and thus, a seamless connectivity of low-power sensors and actuators with higher-value services in networked device infrastructure or the Internet with one comprehensive cross-domain technology. However, these results did not take into account resource requirement analysis and performance evaluations and comparison with existing schemes. Besides, these measurements are done under some specific hypothesis like considering only message sizes as performance metric.

B. Sensor Web Frameworks

Sensor Web is a revolutionary concept towards achieving a collaborative, consistent, and consolidated sensor data collection, fusion and distribution system, typically used in environment monitoring applications. Sensor Webs can act as an extensive monitoring and sensing system that provides timely, comprehensive, continuous and multi-mode observations. This new earth-observation system opens up a new avenue to fast assimilation of data from various sensors and to accurate analysis and informed decision makings.

Sensor Web Frameworks generally aim at making the heterogeneous sensor (and actuator) devices along with sensor reading repository discoverable and accessible from the Internet. In general, they provide a mash-up application that allows visualizing sensory data.

A key challenge in building the Sensor Web is how to automatically access and integrate different types of spatio-temporal data that is observed by sensor devices or generated using simulation models. The *Open Geospatial Consortiums* (OGC) [23] is an international standardization consortium, which provides a framework that specifies standard interfaces to access geographical data in addition to encoding and exchanging these data over the Internet. OGC Web Services follow the W3C's service-oriented web services framework and support publishing, automatically discovering and accessing geographical information over the web; leading to Spatial Data Infrastructures (SDI).

The *Sensor Web Enablement* (SWE) [24] initiative, initiated by the OGC, extends the prominent OGC Web services by providing additional services for integrating Web-connected sensors and sensor systems. The SWE architecture was designed to enable the creation of web-accessible sensor assets through common interfaces and encodings. SWE currently defines four Web service specifications and two models of encodings for observations and sensors, respectively. The four Web services are [24]: (1) *Sensor Observations* Service for requesting, filtering, and retrieving observations and sensor system information, (2) *Sensor Planning* Service for sensor tasking and feasibility studies, (3) *Sensor Alert* Service for publishing and subscribing to alerts from sensors, and (4) *Web Notification* Service as a data transport protocol for the asynchronous delivery of messages or alerts. The two data models and encodings, which are used as data and metadata exchange protocols, are: (1) *Observations & Measurements* (O&M) used for encoding observations and measurements (data) from a sensor device, in real-time and archived modes and (2) *Sensor Model Language* (SensorML), used for describing metadata about sensors systems and processes associated with sensor observations, thus providing information needed for discovery of sensors and the location of sensor observations.

SWE attempts to efficiently address the aforementioned key challenges. In fact, it provides an infrastructure that follows the publish-find-bind paradigm borrowed from the SOA paradigm. It also allows for fusing multiple data models and formats into a common data model and representation. However, SWE has

the limitation that it only provides rudimentary support for the required data conversion.

On the other hand, the SWE framework presents some major gaps when it comes to dynamic data fusion and context-based information extraction. First, SWE does not specify any explicit ontological structure. Although all services make use of a common encoding and transport protocol, semantic interoperability is still an issue due to the lack of an explicit common formal conceptual model encoded in the system (there is only one model in the documentation). The services and encodings focus on providing standard encoding schemes, but they are not grounded in any formal ontology. This impacts on dynamically fusing data with different time granularity, space and measured phenomena. Apart from the lack of semantics, the SWE framework mostly addresses data acquisition but neglects filtering and information overload. Data is basically pulled from passive services rather than being pushed from active services according to the publish-subscribe paradigm. Another drawback of the SWE framework is the lack of support for deploying, discovering and accessing Sensor Web applications. Service providers hide complex application-logic behind OGC services. Users may be aware of individual instances of OGC services, but it will not be clear on exactly what applications each service (or combination of services) supports.

In addition to these standardization efforts, there has been several proprietary solutions that illustrated the Sensor Web concept. For instance, *SensorMap* [25], a Microsoft project, provides a set of tools that data owners can use to easily publish their environmental data, and a GUI enabling users to make queries over live data. SensorMap transparently provides mechanisms to archive, index and aggregate sensory data, and process queries [26]. The SensorMap GUI is a mash-up application that permits users to submit queries on available sensors and overlays the aggregated results on a map. The framework introduced in [27] facilitates access to both real-time and historical sensed data, though of variety of access methods. It addresses the scalability issue by introducing a distributed sensor register.

Although these solutions provide either an SOA-based APIs or common interfaces that make sensing data accessible for the users, their operational behavior is very much influenced by the role of the heavy application-level gateway and single point of failure problem. The application-level gateway play a crucial role due to the absence of direct interaction between sensor nodes, the Internet applications, and users.

IrisNet (Internet-scale Resource-Intensive Sensor Network Services) [28] and *Tenet* [29] are two other approaches that have adopted SOA in developing middleware solutions for WSNs. *Tenet* [29] simplifies application development for tiered sensor networks. It benefits from generic nodes in the lower tier, and masters, which are relatively unconstrained 32-bit platform nodes, in the upper tier. *Tenet* provides a SOA-based solution that, in spite of being flexible to accommodate some applications, is still heavily relying on the application-level gateway that plays an important role in the *Tenet* solution.

On the other hand, *IrisNet* is a distributed software architecture, which provides high-level sensor services to users. It has two main components: (i.) *Sensor Agents* (SA), which provide, pre-process and reduce raw data from a physical sensor, and (ii.) *Organizing Agents* (OA), which provides a sensor service. An Organizing Agent typically collects and analyzes data from Sensor Agents to answer the particular class of queries related to the underlying service. An important characteristic of this architecture is that Sensor Agents are dynamically programmable. The architecture addresses issues that pertain to designing large-scale distributed systems, such as distributed processing, distributed storage and security. Sensory data is typically represented in XML format, and queried are expressed by means of the XPATH query language. Furthermore, *data reduction* is a key concept introduced in this system and consists in extracting higher-level features from raw data from different sensors.

Although these solutions expose WSN to be more accessible through the Internet by means of application-level gateways, they mainly suffer from the single-point-of-failure problem, and scalability issues common to centralized gateway approaches. Additionally, no functionality to enable direct and seamless interaction between wireless sensors and the Internet has been supported, so far.

GeoSwift, another SOA-based framework that was proposed by Liang et al. in [30], is a distributed geospatial information infrastructure for the Sensor Web. This framework consists of a three-layered architecture composed of (i.) the Sensor Layer, which comprises the actual sensor devices, (ii.) the Communication Layer, which represents the physical network or data communication links between the various components, and (iii.) the Information Layer, that ensures interoperability and integration of data among different sensors. The experimental prototype uses a Web services' approach for service registration and discovery. The architecture advocates the use of OGC standards for integrating and exposing sensory data. The authors have proposed an extension to the traditional Web services' approach by relying on a sensing server. The sensing server essentially makes part of the Information Layer, and is able to integrate and store data in different formats from different sensors, and consequently, makes an abstraction of the sensor-specific data formats and communication protocols to the end-user. New sensors can be integrated into the system by extending the sensing server or by deploying a new server.

All aforementioned approaches aim to provide a distributed infrastructure for publishing, discovering and accessing sensor resources and to tackle, to some extent, the challenge of data fusion. They also aim to restrict data access for end-users only to the required information. These approaches are promising for a single or a group of organizations building distributed applications. However, it is questionable whether these approaches will be able to scale well on the Internet, where thousands of different organizations will be developing and deploying different applications comprising trillion of sensor nodes, and much more users will need to tailor and integrate these applications.

C. Some other research efforts

Several other efforts aimed at using SOA in WSNs. In what follows, we present an overview of most relevant works. In [31, 32], the authors addressed the feasibility of using RESTful Web services to integrate SOA with IP-based WSNs. In [31], the authors presented an approach to integrate tiny wireless sensor or actuator nodes into an IP-based network. Sensors and actuators are represented as resources of the corresponding node and are made accessible using a RESTful web service. Sensor nodes run a small web server on top of a TCP/IP stack to provide access to sensor data and actuators using HTTP requests. Data is represented in the JSON format, which is a more lightweight alternative as compared to XML. A prototype application based on TinyOS 2.1 on a custom sensor node platform with 8 Kbytes of RAM and an IEEE 802.15.4 compliant radio transceiver was implemented. A key feature in this approach is that compared to many existing approaches that provide Web services at a smart gateway, it proves the feasibility to provide Web services at each node, even when using a very resource-constrained hardware platform. The system explained in [32] uses two mechanisms to provide a good performance and low-power operation: a session-aware power-saving radio protocol and the use of the HTTP Conditional GET mechanism. In [33], Rezgui and Eltoweissy explored the potential of SOA in building open, efficient, interoperable, scalable, and application-aware Wireless Sensor and Actuator Networks (WSANs). A prototype of service-oriented WSAN was developed using TinyOS. I [34], King et al. developed a service-oriented WSAN platform, called Atlas, which enables self-integrative, programmable pervasive spaces. Kushwaha et al. developed in [35] a programming framework, called OASiS, which provides abstractions for object-centric, ambient-aware, service-oriented sensor network applications. OASiS decomposes specified application behaviors and generates the appropriate node-level code for deployment onto sensor networks. It enables the development of WSN applications without having to deal with the complexity and unpredictability of low-level system and network issues. In [36], Golatowski et al. proposed a service-oriented software architecture for mobile sensor networks. An adaptive middleware is employed in the architecture that encompasses mechanisms for cooperative data mining, self-organization, networking, and energy optimization to build higher-level service structures. In [10], the authors presented an approach to seamlessly integrate WSNs into business process (i.e. SOA) environments using the Business Process Execution Language (BPEL) and Web Services while using only very few resources on the sensor nodes. It introduces how application developers can use standard-compliant techniques to describe business processes that are using services offered by WSNs, without the need for hand-crafted code for data conversion, etc. By adopting this approach, services offered by the WSN can be used seamlessly in enterprise-level business processes. These services can also quickly be composed to higher-level applications by simply modifying the business

process. Priyantha et al. described in [37] a Web Service-based approach based on standard technologies such as IPv6, 6LoWPAN, and HTTP. Considering the message serialization and transport, Web Service messages are exchanged using HTTP. In their approach, they tried to avoid using complex SOAP messages as much as possible. Instead, if Web Service messages do not contain complex data structures, simple URL encoded messages are exchanged to reduce the message size. In [38], Amundson et al. presented a SOA-based approach for WSNs not relying on Web services. Thus, to enable sensor nodes calling Web Services of Enterprise-IT systems, their solution imposes the use of a gateway, which converts between the proprietary middleware message format and standard Web Service message format.

V. CONCLUSIONS AND RESEARCH ROADMAP

In this paper, we have presented a survey on research efforts for the integration between Service-Oriented Architecture and LoWPANs. Certainly, these efforts have clearly contributed to illustrate the SOA/LoWPAN coupling through different and diverse approaches. However, the coupling between SOA and LoWPANs (such as WSNs) still presents several challenges to be investigated, including but not limited to:

- *Adaptation*: The adaptation challenge consists in adapting the concept of SOA of the traditional Internet to the requirements of Cyber-Physical Systems and networks of cooperating objects. Indeed, SOA can provide domain-independent solutions for discovery, selection, composition or aggregation of services to provide meaningful functionality that highly rely on semantics, formal specifications and reasoning. However, SOA architectural principle cannot be used directly. It has to be adapted to additionally include, not only large complex services, but also simple ones, like data storage, routing or sensor readings. It must also be compliant with the requirements of low power and limited resource devices. Identifying and specifying services are crucial for exploiting SOA in WSANs. A large number of questions need to be answered in this respect. For example, how many categories of services should be classified in the context of WSAN? What are the functionalities, interfaces, and properties of each service? What are its quality levels relevant to performance requirements? In particular, how to deal with the difference between sensors and actuators when specifying services?
- *Quality-of-Service*: SOA enables the discovery, access and sharing of the services, data, computational and communication resources in the network by several and different users. It also allows for rapid and cost-effective composition of interoperable and scalable systems based on reusable services exposed by these systems. This is particularly useful for QoS provisioning in Wireless Sensor and Actuators Networks that are integrated into large-scale Cyber-Physical Systems in which multiple applications run on diverse technologies and platforms. The provision of QoS that pertains to SOA over WSNs

is of a paramount importance. Real-time guarantee is one of the main challenges as it is important to deliver services of high priority with a bounded delay and a certain level of reliability. The choice of SOA technology would have a great impact on the QoS and performance, for that reason comparative studies between existing SOA mechanisms would be of a great interest and importance to understand the advantages and limitations of each technique.

- *Composition and Monitoring*: SOA suits particularly well for monitoring systems since the development of the whole network can directly be mapped to the service, simple or complex. For example, the network itself provides several composite high-level services such as area monitoring or manipulation of actuators. In addition, each node offers complex services like data forwarding or sensor readings. Each node can also be represented as a collection of services that interact with each other. In this context, the core challenges lie in devising a communication-level and application-level architecture for monitoring systems that satisfies both SOA and WSNs requirements at the same time. In this respect, several questions arise such as for instance: what sort of service categories should be classified in the context of monitoring systems in general and those compliant with 6LoWPAN in particular? How to adapt the legacy services of the Internet to the requirements of 6LoWPAN networks?
- *Semantics*: Semantic technologies are often proposed as important components of heterogeneous, dynamic information systems. The requirements and opportunities arising from the rapidly growing capabilities of networked sensing devices are a challenging. In this context, one promising research objective would be to develop an understanding of the ways semantic web technologies, including ontologies, agent architectures and semantic web services, can contribute to the growth and the deployment of large-scale sensor networks and their applications. several emerging issues could be investigated, including:
 - *Standardization*: This issue plays an essential role with respect to interoperability. The specification of standards for communication (MAC, routing, topology control, etc.), data representation, service description, service discovery, etc. that cope with LoWPANs constraints is must for supporting large-scale deployment and interoperability of these technologies. Today, some efforts have addressed this issue, such the IEEE 802.15.4, 6LoWPAN, OGC, and ROLL, etc. however, there is a need that all these standardization committee work closely for defining a universal solutions at different levels.
 - *SWS management inside lowPANs*: SWS deals with service publishing, discovery, composition services, invoking, and monitoring. All these services must have quality of service, fault-tolerance, security, dependability, etc. then how to fit all of those in such

constrained environment such as WSNs.

- *Software development process*: Currently, we have a lack of availability of software development process for WSN applications. Yet, it is needed that such process must deal with WSN application development and integration to the whole third-party system.

REFERENCES

- [1] Z. Shelby and C. Bormann, *6LoWPAN: The Wireless Embedded Internet*. Wiley Publishing, 2010.
- [2] J. W. Hui and D. E. Culler, "IP is Dead, Long Live IP for Wireless Sensor Networks," in *SenSys'08: Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*. New York, NY, USA: ACM, 2008, pp. 15–28.
- [3] C. Pautasso and E. Wilde, "RESTful Web Services: Principles, Patterns, Emerging Technologies," in *WWW '10: Proceedings of the 19th international conference on World wide web*. New York, NY, USA: ACM, 2010, pp. 1359–1360.
- [4] C. Pautasso, O. Zimmermann, and F. Leymann, "RESTFUL Web Services vs. "BIG" Web Services: Making the Right Architectural Decision," in *WWW*, 2008, pp. 805–814.
- [5] J. D. Case, M. Fedor, M. L. Schoffstall, and J. Davin, "Simple Network Management Protocol (SNMP)," United States, 1990.
- [6] S. Thomson and T. Narten, "IPv6 Stateless Address Autoconfiguration," United States, 1998.
- [7] M. Geoff, "The 6LoWPAN Architecture," in *EmNets '07: Proceedings of the 4th workshop on Embedded networked sensors*. New York, NY, USA: ACM, 2007, pp. 78–82.
- [8] "The Internet Engineering Task Force." [Online]. Available: <http://www.ietf.org>
- [9] C. Werner, C. Buschmann, and T. Jäcker, "Enhanced Transport Bindings for Efficient soap Messaging," in *ICWS*, 2005, pp. 193–200.
- [10] N. Glombitza, D. Pfisterer, and S. Fischer, "Integrating Wireless Sensor Networks into Web Service-Based Business Processes," in *MidSens '09: Proceedings of the 4th International Workshop on Middleware Tools, Services and Run-Time Support for Sensor Networks*. New York, NY, USA: ACM, 2009, pp. 25–30.
- [11] A. Scholz, I. Gaponova, S. Sommer, A. Kemper, A. Knoll, C. Buckl, J. Heuer, and A. Schmitt, "eSOA - Service-Oriented Architectures Adapted for Embedded Networks," in *Proceedings of 7th IEEE International Conference on Industrial Informatics*, June 2009, pp. 599–605.
- [12] F. Jammes and H. Smit, "Service-Oriented Paradigms in Industrial Automation," *Industrial Informatics, IEEE Transactions on*, vol. 1, no. 1, pp. 62–70, April 2005. [Online]. Available: <http://dx.doi.org/10.1109/TII.2005.844419>
- [13] L. M. S. de Souza, P. Spiess, D. Guinard, M. Khler, S. Karnouskos, and D. Savio, "SOCRADES: A Web

- Service Based Shop Floor Integration Infrastructure,” in *IOT*, ser. Lecture Notes in Computer Science, vol. 4952. Springer, 2008, pp. 50–67.
- [14] P. Costa, G. Coulson, and C. Mascolo, “The runes Middleware: A Reconfigurable Component-Based Approach to Networked Embedded Systems,” in *In Proc. of 16th International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC05)*. IEEE Press, 2005, pp. 11–14.
- [15] M. Kushwaha, I. E. Amundson, X. Koutsoukos, S. Neema, and J. Sztipanovits, “OASiS: A Programming Framework for Service-Oriented Sensor Networks,” in *IEEE/Create-Net COMSWARE 2007*, January 2007.
- [16] Devices Profile for Web Services. [Online]. Available: <http://www.opengeospatial.org/projects/groups/sensorweb>
- [17] H. Abangar, P. Barnaghi, K. Moessner, A. Nnaemego, K. Balaskandan, and R. Tafazolli, “A Service Oriented Middleware Architecture for Wireless Sensor Networks,” in *Proceedings of the Future Network & Mobile Summit 2010 Conference*, 2010.
- [18] F. Jammes, A. Mensch, and H. Smit, “Service-Oriented Device Communications Using the Devices Profile for Web Services, booktitle = MPAC ’05: Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing, year = 2005, isbn = 1-59593-268-2, pages = 1–8, location = Grenoble, France, publisher = ACM, address = New York, NY, USA,,”
- [19] G. Moritz, E. Zeeb, S. Prüter, F. Golatowski, D. Timmermann, and R. Stoll, “Devices Profile for Web [s]ervices in Wireless Sensor Networks: Adaptations and Enhancements,” in *ETFA’09: Proceedings of the 14th IEEE international conference on Emerging technologies & factory automation*. Piscataway, NJ, USA: IEEE Press, 2009, pp. 43–50.
- [20] I. K. Samaras, J. V. Gialelis, and G. D. Hassapis, “Integrating Wireless Sensor Networks into Enterprise Information Systems by Using Web Services,” in *SENSORCOMM ’09: Proceedings of the 2009 Third International Conference on Sensor Technologies and Applications*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 580–587.
- [21] G. Moritz, D. Timmermann, R. Stoll, and F. Golatowski, “Encoding and Compression for the Devices Profile for Web Services,” in *AINA Workshops*, 2010, pp. 514–519.
- [22] —, “encDPWS - Message Encoding of SOAP Web Services,” in *PerCom Workshops*, 2010, pp. 784–787.
- [23] Open Geospatial Consortium. [Online]. Available: <http://www.opengeospatial.org/>
- [24] Sensor Web Enablement WG. [Online]. Available: <http://www.opengeospatial.org/projects/groups/sensorweb>
- [25] S. Nath, J. Liu, and F. Zhao, “Sensormap for Wide-Area Sensor Webs,” *Computer*, vol. 40, no. 7, pp. 90–93, 2007.
- [26] A. Santanche, S. Nath, J. Liu, B. Priyantha, and F. Zhao, “SenseWeb: Browsing the Physical World in Real time,” in *Demo Abstract*, April 2006.
- [27] H. M. I. Rhead, M. Merabti and P. Fergus, “Worldwide Sensor Web Framework Overview,” in *Proceedings of the 9th Annual Postgraduate Symposium, The Convergence of Telecommunications, Networking and Broadcasting*, 2008.
- [28] P. B. Gibbons, B. Karp, Y. Ke, S. Nath, and S. Seshan, “IrisNet: An Architecture for a Worldwide Sensor Web,” *IEEE Pervasive Computing*, vol. 2, pp. 22–33, 2003.
- [29] J. Paek, B. Greenstein, O. Gnawali, K.-Y. Jang, A. Joki, M. Vieira, J. Hicks, D. Estrin, R. Govindan, and E. Kohler, “The Tenet Architecture for Tiered Sensor Networks,” *ACM Trans. Sen. Netw.*, vol. 6, no. 4, pp. 1–44, 2010.
- [30] S. H. L. Liang, A. Croitoru, and C. V. Tao, “A Distributed Geospatial Infrastructure for Sensor Web,” *Comput. Geosci.*, vol. 31, no. 2, pp. 221–231, 2005.
- [31] L. Schor, P. Sommer, and R. Wattenhofer, “Towards a Zero-Configuration Wireless Sensor Network Architecture for Smart Buildings,” in *BuildSys ’09: Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*. New York, NY, USA: ACM, 2009, pp. 31–36.
- [32] D. Yazar and A. Dunkels, “Efficient Application Integration in IP-based Sensor Networks,” in *BuildSys ’09: Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*. New York, NY, USA: ACM, 2009, pp. 43–48.
- [33] A. Rezgui and M. Eltoweissy, “Service-Oriented Sensor-Actuator Networks: Promises, Challenges, and the Road Ahead,” *Computer Communications*, vol. 30, no. 13, pp. 2627–2648, 2007.
- [34] J. King, R. Bose, null Hen-I Yang, S. Pickles, and A. Helal, “Atlas: A service-oriented sensor platform: Hardware and middleware to enable programmable pervasive spaces,” *Local Computer Networks, Annual IEEE Conference on*, vol. 0, pp. 630–638, 2006.
- [35] M. Kushwaha, I. Amundson, X. Koutsoukos, S. Neema, and J. Sztipanovits, “OASiS: A Programming Framework for Service-Oriented Sensor Networks,” in *In IEEE/Create-Net COMSWARE 2007*, 2007.
- [36] F. Golatowski, J. Blumenthal, M. H. M. Haase, H. Burchardt, and D. Timmermann, “Service Oriented Software Architecture for Sensor Networks,” in *In Proc. Int. Workshop on Mobile Computing (IMC03)*, 2003, pp. 93–98.
- [37] N. B. Priyantha, A. Kansal, M. Goraczko, and F. Zhao, “Tiny Eeb Services: Design and Implementation of Interoperable and Evolvable Sensor Networks,” in *SenSys ’08: Proceedings of the 6th ACM conference on Embedded network sensor systems*. New York, NY, USA: ACM, 2008, pp. 253–266.
- [38] I. Amundson, M. Kushwaha, X. Koutsoukos, S. Neema, and J. Sztipanovits, “Efficient Integration of Web Services in Ambient-Aware Sensor Network Applications,” in *3rd IEEE/CreateNet International Workshop on Broadband Advanced Sensor Networks (BaseNets 2006)*, October 2006. [Online]. Available: <http://chess.eecs.berkeley.edu/pubs/284.html>