Distributed Semantic Discovery for Web-of-Things Enabled Smart Buildings

Gérôme Bovet*‡ and Jean Hennebert†‡
*LTCI
Telecom ParisTech, 46 Rue Barrault, 75013 Paris, France
Email: gerome.bovet@telecom-paristech.fr
†DIUF
University of Fribourg, Bd de Pérolles 90, 1700 Fribourg, Switzerland
Email: jean.hennebert@unifr.ch
‡iCoSys
University of Applied Sciences Western Switzerland, Bd de Pérolles 80, 1700 Fribourg, Switzerland

***Abstract*—Nowadays, our surrounding environment is more and more scattered with various types of sensors. Due to their intrinsic properties and representation formats, they form small islands isolated from each other. In order to increase interoperability and release their full capabilities, we propose to represent devices descriptions including data and service invocation with a common model allowing to compose mashups of heterogeneous sensors. Pushing this paradigm further, we also propose to augment service descriptions with a discovery protocol easing automatic assimilation of knowledge. In this work, we describe the architecture supporting what can be called a Semantic Sensor Web-of-Things. As proof of concept, we apply our proposal to the domain of smart buildings, composing a novel ontology covering heterogeneous sensing, actuation and service invocation. Our architecture also emphasizes on the energetic aspect and is optimized for constrained environments.**

***Keywords*—Smart buildings, Discovery, Semantics, Ontologies**

I. Introduction

With the evolution achieved in pervasive computing during the last years, sensors, actuators, mobile devices and smart appliances have become ubiquitous. We are nowadays able to find them in many application scenarios, ranging from small wireless sensor networks like weather monitoring up to big-sized networks like smart buildings and even smart cities composed of ten thousands of such devices. Due to their dynamics, devices can appear and disappear in a short period of time, move to another location and even be reassigned to other purposes. In this context, composing physical-centred applications represents a quite huge challenge as developers must first gather knowledge about what the physical world can offer to them. This problematic also applies to machine-to-machine communications where physical mashups are realized by devices themselves with no prior knowledge or human intervention. For example, a motion sensor could discover some eligible partners to form a specific use case as someone enters a room by switching lights, moving the blinds and activating the heating. Such a behaviour not only increases the human comfort but also participates to the ecological and economical trends of saving energy in buildings. To attain such a strategy, a service and discovery infrastructure is needed, connecting sensors and actuators to form an homogeneous overlay where they will communicate their capabilities and intentions in well-understood machine-processable formats. Reaching such an interaction style thus allows an interconnection between various manufacturers and different device types.

Up to now, the worlds of the Web and sensor networks, especially building automation systems (BAS) were isolated from each other. As a consequence of this segregation, automatic discovery is difficult to achieve, and requires a human in the loop to find, aggregate, and use information from both worlds. The HTML language and HTTP protocol appear as key-enablers of an homogeneous application layer relying on different hardware platforms and software components. Augmenting sensors and BAS with Web capabilities in form of lightweight Web services ensuring interoperability has already been proposed for the well-spread KNX and Enocean building automation standards [1]. However, those technologies, while relying on Web service standards, do not cover the automation of discovery, composition and invocation of Web services [2]. To solve those shortcomings, the Semantic Web [3] pushes open technologies for enabling operations with as less human intervention as possible. Although this concept paved the way for semantic augmented things, some issues relative to the dynamic aspect of sensor networks and to their energy efficiency remain open.

In this paper, we propose a fully distributed Web service discovery architecture that is designed to be robust, reliable and energy efficient. It relies on the paradigm of the Web-of-Things (WoT) where every device capability including discovery is represented as a Web resource [4]. Following this approach makes our discovery architecture loosely-coupled, providing primitives for publishing and discovering Web services. From a scalability and adaptability point of view, our proposed architecture can be adapted to any kind of ontologies and even description languages. In order to improve interoperability between devices, we developed an ontology not only describing device capabilities but also method invocation mechanisms. The descriptions are presented by following the RDF best practices [5]. We argue this choice by leveraging on its semantic richness allowing an automatic composition of Web services in M2M applications. Working in the context of smart

This paper has been accepted for publication in the proceedings of SmartCity'2014 Workshop that had been held in conjunction with NTMS'2014.

buildings where the final aim is to save energy, a particular attention is given to the energetic impact of our discovery architecture.

This paper is organized as follows. The next section refers and summarizes related works. In Section III, we provide a list of requirements that a discovery architecture for smart buildings must fulfil. Section IV introduces the ontology we composed for improving interoperability. The distributed Web service discovery architecture is described in Section V. A reference implementation is provided in Section VI. Finally, Section VII concludes our paper and provides insights on further research.

II. Related Work

Many XML schemas and other data models were developed trying to describe processes and physical environments [6]. The Semantic Sensor Network Ontology (SSN) is based around concepts of systems, processes and observations [7]. It supports the description of the physical and processing structure of sensors. However notions of units and locations are not part of it but can be completed by including other ontologies like the DOLCE Ultra Lite (DUL) [8]. A concrete application of semantics to building automation systems has been proposed in [9].

Augmenting simple Web semantics with discovery capabilities has already been explored in several projects. The notion of shared space centralizing RDF descriptions of a logical entity was introduced in [10]. Queries are performed by the client providing RDF triples in a HTTP request following the REST architectural style that will be applied to the RDF descriptions of the Web services. Pushing this concept further, *SPITFIRE* takes advantages of SPARQL, a language especially conceived for querying RDF descriptions [11][12]. Unicast CoAP discovery requests are sent to a repository where RDF descriptions are stored [13]. This approach requires a service announcement mechanism by service providers. It also offers a possibility to look up for devices that are in a particular state. In this approach, dynamic properties like state values are probabilistically inferred from past data. Although it provides a way for filtering Web services according to dynamic properties, uncertainty remains about the real state so that the query could respond with false positives. Our work is based on the concepts introduced by SPITFIRE. We enhance it with a new ontology describing services invocations. We also complete it with a new architecture taking account of real state values instead of predictions. Finally, we propose best practices for limiting the number of network packets and thus reduce the overall energy consumption.

III. Requirements for Smart Buildings

For building automation systems an extension from simple discovery to query is required. We can list a minimal set of requirements for a discovery architecture in the context of smart buildings:
(1) Optimized for constrained devices - Smart buildings are composed of IP-enabled sensors and actuators relying on electronics offering only few computational power. Other equipments based on KNX, BACnet or EnOcean technologies are typically not IP and will require light gateways to expose the services in the IP world.
(2) Plug-and-play installation of devices - Devices should be integrated in the existing architecture with no human interaction.
(3) Discovery of the entire network - For management purposes, an overview of the entire network should be retrievable in a simple manner.
(4) Selection of devices according to some contextual parameters - Contrarily to classic pervasive environments where only static parameters are needed for looking up specific services, a more dynamic approach is necessary in the context of smart buildings. This aspect is further explained in this section.
(5) Scalability and fault tolerance in energy efficient context - A scalable architecture is required as networks in smart buildings will evolve over time, including new devices, themselves running under new technologies. Fault tolerance is also required considering the applications in buildings such as access, lighting, heating, etc. Such requirements are further discussed in the context of having an energy efficient technology.

A. Web Service Discovery Models

Several discovery models already exist and can be applied to our scenario. We here depict the most widespread ones and argue which one should be used for smart buildings.

Static service discovery is available since the beginning of the Web. In this scenario, descriptions are stored locally on the endpoints and do not change over time. In the *centralized* discovery approach, discovery repositories store descriptions announced and published by endpoints. Discovery requests are sent to the repositories instead of the service endpoints. The model where a distributed approach is used rather than central repositories is called *dynamic*. Infrastructures following this approach are usually leveraging on P2P, data federation or are agent-based [14]. In order to fulfil requirement 5, we opted for a decentralized model in our infrastructure. This choice is argued by the various features that are provided by this model, such as scalability, reliability, and no single point of failure.

Regarding the filtering stage, it can be performed at two different locations. A first naive manner is to empower clients with this responsibility. Any device present on the network will respond with its description to the client, that will in a second step process all responses to find matches. The second approach can be considered as more optimized in terms of network traffic, as query processing is delegated to the service endpoints. Only Web services having a description matching with the query parameters will respond to the client. In our architecture, we decided to follow the latter option as it is compliant with requirement 5 concerning the energetic impact.

B. Concept of a Sensor's Context

The context of a device or a Web service is decisive during discovery. Many building management systems have to look for devices being in a certain state or context for regulation or alarming purposes. For example a building management system (BMS) could search for temperature sensors on the ground floor having measured in the last five minutes a value above 25°C. This request must also consider devices that have

not yet reported and are not known by the system. Relying on a plug-and-play approach allows to have highly dynamic systems requiring no previous knowledge of the available resources. We introduce here the concept of static and dynamic resources properties as illustrated in Fig. 1, which allows us to comply with requirement 4. Static properties will not or rarely vary over time, while dynamic ones are subject to frequent changes.

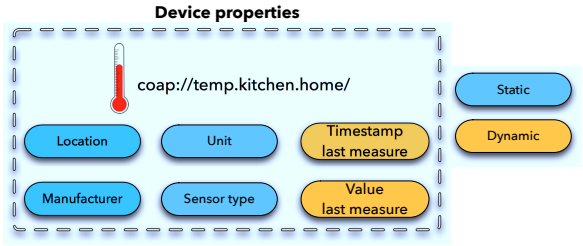

Fig. 1. Static and dynamic properties of a resource

IV. The Ontology Model

As previously mentioned, the SPITFIRE project proposes a SSN based ontology for describing devices properties. Although the ontology includes many aspects for describing intrinsic sensor properties, the related Web services for interacting with devices capabilities are not covered. This situation leads to incompatibilities between devices as a client has no clue about how to consume a Web service. Let us illustrate this with a switch that has to bind with a relay for controlling lighting in a room. Following the proposed discovering principle in SPITFIRE, it will be able to know that there is a light relay in the specific room. However this is not enough when composing mashups as the client should also have knowledge about the Web service's properties (e.g. transport protocol, allowed parameters and their range, return value, etc.). Based on this observation, we can argue that an ontology should also include knowledge for describing properties of Web services and how to consume them. We therefore propose a new ontology that we motivate in the remainder of this section. An example of our proposal is given in Fig. 2.

A. Device Properties

The device properties that are responding to the question "*what does the resource offer?*" are split into three different categories as follows: (1) Sensor types are expressed using the SSN ontology and extended to include any device type that can be present in a building automation system (represented in blue). (2) The location information is indicated by relying on the DUL ontology that provides a vocabulary for expressing components of a building (i.e. room, floor, building, etc.) (represented in orange). (3) The value types and formats are described by selecting the appropriate datapoint (represented in pink). In BAS, datapoints represents endpoints for a specific value.

B. Gates

The description of the procedure to consume a Web service is given through gates (represented in green). We here rely on the concept of RESTdec proposing a semantic facilitating

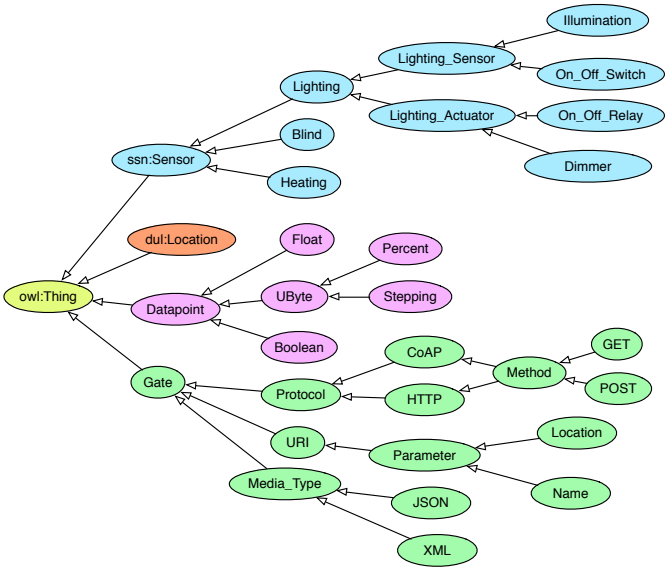

Fig. 2. Excerpt of the proposed ontology dedicated to building automation systems

service consumption between devices [15]. Each service owns a description of which protocols it supports (e.g. HTTP, CoAP, FTP, etc.), allowing a client to select the most appropriate one. The method that has to be used for consuming the service is also given by the ontology. The URI for accessing the service is decomposed in multiple parts especially regarding the parameters. Their name and location inside the URI are specified. Finally, the media type of the payload and response are defined by the ontology which allows a client to have knowledge about how to parse a response. Those properties are sufficient for a client to automatically compose a requests and thus allows an inter-compatibility between various technologies.

V. Semantic WoT Discovery and Composition

Following the paradigm of the Web-of-Things, each device represents its capabilities in terms of Web resources that are accessible though a RESTful API. As this approach comes with many advantages like being loosely-coupled and compatible with current Web standards, there is only a small footstep from enhancing the WoT with semantic discovery. In this Section we describe the main three components necessary for building a semantic WoT, which are (1) augmenting sensors with semantic descriptions, (2) providing a querying mechanism for retrieving knowledge, and finally (3) automatically building mashups between different device types.

A. Sensor Semantics

In a vision of a semantic Web, each resource has to be described with semantics in order to achieve interoperability. As previously explained, this is realized by forming RDF descriptions using a vocabulary provided in a shared ontology. Listing 1 shows a RDF description excerpt (only the device property part) of a temperature sensor. When considering a resource providing only static properties, the RDF description can be built at resource creation and will remain valid until the resource stops existing. Regarding resources that involve

dynamic properties like it is the case for most sensors, the RDF description has to be up-to-date with the actual status of the resource. Instead of holding internally a RDF document that has to be updated each time a dynamic property changes, we propose to build this document upon request when a client retrieves the description. This way of functioning lessens the required memory space and computing power.

Listing 1. Example of a temperature sensor device properties description using RDF with the Semantic Sensor Network Ontology

```
@prefix rdf: <http://www.w3.org/1999/02/22−rdf−syntax−ns#> .
@prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn> .
<coap://temp.kitchen.home>
  rdf:type ssn:SensingDevice ;
  ssn:observes <http://purl.oclc.org/NET/muo/ucum/physical−
       quality/temperature> ;
  ssn:hasMeasurementCapability <coap://temp.kitchen.home/temp> .
<coap://temp.kitchen.home/temp>
  rdf:type ssn:MeasurementCapability ;
  ssn:hasMeasurementProperty <coap://temp.kitchen.home/hum/accuracy> ;
  ssn:hasMeasurementProperty <coap://temp.kitchen.home/hum/sensivity> .
<coap://temp.kitchen.home/hum/accuracy>
  rdf:type ssn:Accuracy ; rdf:value 1 .
<coap://temp.kitchen.home/hum/sensivity>
  rdf:type ssn:sensivity ; rdf:value 0.1 .
```

Assuming a client already knows the existence of a Web resource or has just discovered it, it needs to retrieve the RDF description in order to be able to understand what it provides and how to communicate with it. Different scenarios can be conceived depending on the application protocol used. A first approach consists of using a special code in the request indicating one wants to access the description of the mentioned resource. We can illustrate this principle when using HTTP by setting the method to OPTION, as proposed by [15]. Unfortunately the CoAP protocol does not provide the OPTION method. Additionally, CoAP having a very tightened header, there is no possibility to set a specific flag inside a request. To cope with this problematic, we consider RDF documents as a sub-resources that can be accessed with a GET request. By putting a special placeholder at the end of the URL pointing to the Web resource, one indicates its intention to retrieve the related description.

B. Querying for Resources

Having linked Web resources with RDF semantic description, discovery agents are now able to query resources. To comply with requirements 2, 3 and 5, we opted for a decentralized architecture leveraging on multicast queries. Using multicast for sending queries through the network allows, with no prior knowledge of the infrastructure, to reach any device providing descriptions. Furthermore, multicast reduces the energy impact as only one packet is sent by the discovery agent, whereas other discovery architectures based on unicast will have much more impact on the energetic aspect. Only sensors having resources matching the query will respond to the request, which also limits the network traffic.

As every WoT device embeds a Web server, we enhance it with a new service that will be responsible for responding to discovery queries. Any device offering discovery capabilities will thus expand its API (e.g. coap://233.67.1.26/discover) with a new service bond to the multicast group which is either pre-configured or obtained from an auto-configuration system.

Finally, the discovery agent will express the query using the SPARQL language that will be placed as payload inside the request. Queries can range from very simple ones resulting in the discovery of the entire network or contain restraining properties which will limit the scope of results. A major advantage of combining RDF documents and SPARQL is that a client can specify what kind of property it wants as result (e.g. the URL to the resource, the actual value, the device model, etc.). This allows a new interaction style where, using only one request, different kind of return values can be expected. For example, a building management system could look for rooms where the temperature is above a certain value by specifying in the SPARQL that it wants to have only the location of the sensor as result and does not care about the URL for accessing the resource or other properties. Listing 2 provides an example of a query seeking for temperature sensors located in the kitchen.

Listing 2. Example of a SPARQL query to look after temperature sensors with a value higher than 25° Celsius located in the kitchen

```
@prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn> .
SELECT ?node WHERE {
  ?node a ssn:Sensor ;
    ssn:observes <http://purl.oclc.org/NET/muo/ucum/physical−
         quality/temperature> .
    dul:hasLocation "kitchen" ; rdf:value ?lv ; FILTER (?lv > 25) .
}
```

C. Automatic Composition of Mashups

The ultimate aim of semantics is to automate interaction and composition between various applications. In the same perspective, sensors can benefit from semantics by being able to discover potential partners and to build mashups automatically with no human in the loop. Since our proposed ontology not only describes resources properties but also so-called gates, we open the path for a new self-composing sensor network. In classical approaches, administrators or developers are required to manually link devices with each other following a pre-configured and non-scalable interaction mechanism (i.e. chose between HTTP or CoAP, available parameters, their position and meaning). From now on, scenarios where a new sensors can integrate itself in an existing environment by discovering neighbours are possible. For example, an illumination sensor being responsible for the whole lighting will send a query to discover potential partners in the same room. Since each device describes its gates, the illumination sensor will know how to register for presence notifications on the presence sensor and how to turn the light on or off and also to drive the blinds.

VI. Implementation

In order to demonstrate the feasibility of our proposed architecture, we developed a prototype implementation based on Java technologies running on several Raspberry Pi. The scenario introduced in Section V-C served for validating our approach of automatic mashup composition. The architecture of our reference implementation that is further depicted in this Section is illustrated in Fig. 3.

We opted for jcoap as base implementation for CoAP [16]. As at the time of writing this paper no implementation offered multicast support (CoAP group communication), we enhanced jcoap with this new functionality for the server and client

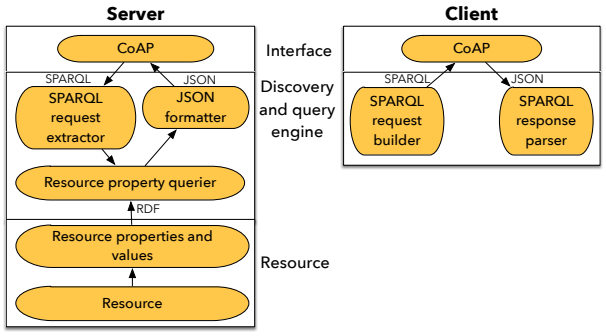

Fig. 3. Modules composing the client and server discovery architecture

sides. The CoAP server listens on a multicast socket with preconfigured port and IP address. Its role is to dispatch incoming discovery requests to the discovery and query engine. If matches are found, it will respond to the source of the request with the according payload data.

Going one level down in the server, the discovery and query engine is responsible for evaluating requests and finding corresponding matches. It starts with a SPARQL request extractor that will take out the query from the CoAP messages and perform some validations regarding the format. Once the request is considered as well-formed, it is passed to the querier. This module will gather the RDF representations of all the resources present and available on the device. Once the collection complete, the SPARQL query is applied to each RDF representation of the collection. Each match is then stored in a collection of results. If the collection of results is empty after querying, the process can stop at this step. Otherwise, the collection is forwarded up to the JSON formatter. This module iterates through the results collection and formats SPARQL responses to JSON. The last step consists of responding to the client over the CoAP interface with the results. All this part is handled by using the OpenRDF Sesame framework providing a Java API for processing SPARQL queries [17].

Finally, the Web resource has to provide its entire description in form of a RDF document. Each time the description is retrieved by either a remote client or internally by the query engine, this module will collect the actual state of dynamic properties and place them inside the RDF document. According to a format parameter value, the description can either be returned in standard XML, JSON or Turtle N3.

VII. Conclusion and Future Works

It is likely that the combination of ontologies and semantics will play a key role in the field of sensor networks. They indeed pave the way to a homogeneous global inter-compatible network whereas current installations form small islands isolated from each other. The main advantage of semantic descriptions resides in being resource oriented which fits with the concept of the Web-of-Things where every device capability is represented as a Web resource accessible over a RESTful API. Semantically augmenting things not only exposes their properties to the rest of the world but also allows a new query mechanism facilitating mashup composition. The need for a human in the loop is greatly reduced, converging to totally autonomous networks.

In this paper we showed our vision of semantics applied to sensor networks with a particular emphasis on smart buildings. Unlike the approach taken in SPITFIRE where dynamic properties are inferred from statistical data, our system considers the real state of a property avoiding false positives. Our enhanced ontology adds the vocabulary for describing gates and thus enables an automatic consumption of services. Future developments intends to make the architecture compatible with very constrained devices that can not run a Java application and also to even more reduce the energy impact. The security aspect is also an important concern that has to be solved in the near future.

Acknowledgment

The authors are grateful to the Swiss Hasler Foundation and to the RCSO grants from the HES-SO financing our research in this exciting area of smart buildings.

References

[1] G. Bovet and J. Hennebert, "Offering web-of-things connectivity to building networks," in *Proc. of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp 2013)*, 2013.

[2] B. Sapkota, L. Vasiliu, I. Toma, D. Roman, and C. Bussler, "Peer-to-peer technology usage in web service discovery and matchmaking," in *Proc. of the 6th International Conference onWeb Infromation Systems Engineering (WISE 05)*, 2005.

[3] T. Bermers-Lee, J. Hendler, and O. Lassila, *The Semantic Web*. Scientific American, 2001.

[4] D. Guinard, "A web of things application architecture – integrating the real-world into the web," Ph.D. dissertation, ETHZ, 2011.

[5] G. Klyne and J. Caroll, "Resource description framework (rdf): Concepts and abstract syntax," World Wide Web Consortium, Tech. Rep., 2004.

[6] W. S. S. N. I. Group, "Review of sensor and observation ontologies," http://www.w3.org/2005/Incubator/ssn/wiki/Incubator_Report\#Review_of_Sensor_and_Observation_ontologies.

[7] ——, "Semantic sensor network ontology," http://www.w3.org/2005/Incubator/ssn/ssnx/ssn.

[8] "Dolce ultralite ontology," http://ontologydesignpatterns.org/wiki/Ontology\%3aDOLCE\%2BDnS_Ultralite.

[9] M. Jung, J. Weidinger, C. Reinisch, W. Kastner, C. Crettaz, A. Olivieri, and Y. Bocchi, "A transparent ipv6 multi-protocol gateway to integrate building automation systems in the internet of things," in *Proc. of the IEEE International Conference on Green Computing and Communications*, 2012.

[10] B. Sapkota, D. Roman, S. Kruk, and D. Fensel, "Distributed web service architecture," in *Proc. of the International Conference on Internet and Web Applications and Services (AICT-ICIW 2006)*, 2006.

[11] D. Pfisterer and K. Romer, "Spitfire: toward a semantic web of things," *Communication Magazine*, vol. 49, 2001.

[12] E. Prud'hommeaux and A. Seaborne, "Sparql query language for rdf," World Wide Web Consortium, Tech. Rep., January 2008.

[13] Z. Shelby, K. Hartke, and C. Bormann, "Constrained application protocol (coap)," IETF, http://tools.ietf.org/html/draft-ietf-core-coap-18, Tech. Rep. 18, 2013.

[14] F. Zhu, M. Mutka, and L. Ni, "Service discovery in pervasive computing environments," *IEEE Pervasive Computing*, vol. 4, no. 4, pp. 81–90, 2005.

[15] R. Verborgh, T. Steiner, D. V. Deursen, R. van de Walle, and J. Vallés, "Efficient runtime service discovery and consumption with hyperlinked restdec," in *Proc. of the 7th International Conference on Next Generation Web Services Practices (NWeSP 2011)*, 2011.

[16] "jcoap," http://code.google.com/p/jcoap/.

[17] "Openrdf sesame," http://www.openrdf.org/.