Service Oriented Middleware for the Internet of Things: A Perspective*

(Invited Paper)

Thiago Teixeira, Sara Hachem, Valérie Issarny, and Nikolaos Georgantas

INRIA Paris-Rocquencourt, France

Abstract. The Internet of Things plays a central role in the foreseen shift of the Internet to the Future Internet, as it incarnates the drastic expansion of the Internet network with non-classical ICT devices. It will further be a major source of evolution of usage, due to the penetration in the user's life. As such, we envision that the Internet of Things will cooperate with the Internet of Services to provide users with services that are aware of their surrounding environment. The supporting service-oriented middleware shall then abstract the functionalities of Things as services as well as provide the needed interoperability and flexibility, through a loose coupling of components and composition of services. Still, core functionalities of the middleware, namely service discovery and composition, need to be revisited to meet the challenges posed by the Internet of Things. Challenges in particular relate to the ultra large scale, heterogeneity and dynamics of the Internet of Things that are far beyond the ones of today's Internet of Services. In addition, new challenges also arise, pertaining to the physical-world aspect that is central to the IoT. In this paper, we survey the major challenges posed to service-oriented middleware towards sustaining a service-based Internet of Things, together with related state of the art. We then concentrate on the specific solutions that we are investigating within the INRIA ARLES project team as part of the CHOReOS European project, discussing new approaches to overcome the challenges particular to the Internet of Things.

1 Introduction

The Internet of Things (IoT) is characterized by the integration of large numbers of real-world objects (or *things*) onto the Internet, with the aim of turning high-level interactions with the physical world into a matter as simple as is interacting with the virtual world today. As such, two types of devices that will play a key role in the IoT are sensors and actuators.

In fact, such devices are already seeing widespread adoption in the highly-localized systems within our cars, mobile phones, laptops, home appliances, etc. In their current incarnation, however, sensors and actuators are used for little more than low-level inferences and basic services. This is partly due to their highly specialized domains (signal

* This work is supported by the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement number 257178 (project CHOReOS - Large Scale Choreographies for the Future Internet - `http://www.choreos.eu`).

W. Abramowicz et al. (Eds.): ServiceWave 2011, LNCS 6994, pp. 220–229, 2011.

processing, estimation theory, robotics, etc.), which demand application programmers to assume the role of domain experts, and partly due to a glaring lack of interconnectivity between all the different devices.

To be truly useful, sensors and actuators must be ubiquitous rather than constrained to an area around a small set of personal devices, such as a mobile phone or a car. This translates to having a network with a massive number of *things*, spread over a large area — until it covers the entire world. But as the number of sensors and actuators in a network grows to millions and even billions, interoperability, scalability, and flexibility challenges arise. Many of these challenges are, at the surface, the same as those already observed in the existing Internet. However, as we contend later in this paper, these challenges are often significantly different when taking into consideration the complexity of handling physical-world information, especially at never-before-seen scales.

The goal of this paper is to discuss these challenges and propose new directions for solutions at the middleware layer. Throughout this discussion, we take a service-oriented view by abstracting a *thing* as a software service that also has a physical side: that is, *things* sense/actuate the physical world, and they carry physical attributes such as location and orientation in space. We start in Section 2 by stating what in our view are the foremost challenges of the IoT. Then, Section 3 surveys the existing IoT middleware in the literature. Section 4 continues by presenting an overview of our envisioned service-oriented middleware for the IoT, which aims to address the challenges posed by the IoT by leveraging well-studied characteristics of physical phenomena. Finally, Section 5 concludes the paper.

2 Challenges

Many of the challenges related to the Internet of Things are directly inherited from the existing Internet. However, when factoring in the massive scale of the IoT with the intricacies of handling the physical-world information (something that is not considered in the traditional Internet), we find that even some of the most commonly-studied challenges of the Internet appear with significant differences in their IoT manifestation.

In the discussion below, we describe those challenges that stand out, using minimalistic examples in an effort to pinpoint the fundamental cause of each of them. Note, however, that real-world uses of the IoT will no doubt be much more complex. As a result, these challenges will appear not in isolation, but rather as any number of different combination thereof. What we search is a system that can address these even when they occur simultaneously.

1. **Scale**—When performing a sensing or actuation task that pertains to millions of Sensors or Actuators (S&A), it is often infeasible to coordinate every one of the required devices due to constraints such as time, memory, processing power, and energy consumption. To put this into perspective, consider the simple case of an application that requires to know the average air temperature on the city of Paris at this very moment. The answer to this query can be "easily" found by calculating the mean value of the set of temperature readings of *all the thermometer-carrying devices* in the region. However, if there are millions of such devices in Paris, then this set of temperature readings quickly grows unmanageable. Thus even a simple-looking query such as this often leads to unattainable results when the massive

scale of the IoT is factored in. In this example, the solution taken by a domain expert might be to approximate the average temperature by selecting a small sample of temperatures uniformly-distributed within the city, using the well-known equations for the sampling distribution of the mean to calculate the optimal sample size. However, how can an IoT middleware bypass the need for a human expert in the loop, and perform this probabilistic sensor selection on its own?

2. **Deep Heterogeneity**—An important aspect of the IoT that is usually not emphasized enough is that services representing *things* are much more heterogeneous than typical services on the current Internet. For one, due to cost considerations, new sensing/actuating hardware *will often not replace* older generations in already-deployed networks — rather, different generations of devices will operate alongside one another. Likewise, it is probable that the future Internet will be composed of numerous sensor/actuator networks deployed by distinct entities, for distinct reasons. In all of these cases, these networks are bound to contain devices from an assortment of vendors, with highly varying sensing/actuating characteristics, such as error distributions, sampling rates, spatial resolution, and so on. All of these parameters (including functional and non-functional properties) lead to a *deep heterogeneity* that makes S&A networks extremely hard to work with, even for experts. And as networks increase in size, delegating these types of coordination tasks to humans will simply not be feasible. In such a dynamic environment, with so many unknowns, it is clear that fully automated methods for high level inference will become a necessity.
3. **Unknown Topology**—Much like the existing Internet, one of the IoT's main characteristics is the fact that its topology is both unknown and dynamic. As a consequence, applications will often end up depending on services which are not actually available from any single preexisting component of the network at that given time. For instance, if an application would like to obtain the value of the wind-chill factor at a certain location, it may happen that the network does not have a wind-chill sensor in that exact neighborhood. However, if instead the network does have temperature and wind speed sensors (i.e. anemometer), then a field expert could easily obtain the desired information through the composition of the temperature and wind speed readings using the well-known wind-chill equation. This is possible because the *function* of a wind-chill sensor is equivalent to the *function* provided by the thermometer/anemometer combination put together by the expert. The question is, then: can an IoT middleware perform these types of functional substitutions on its own without supervision? How can this type of service composition be performed in an optimal manner when the network is massive in scale, with unknown topology?
4. **Unknown Data-Point Availability**—A second consequence of the unknown topology is that sometimes there will be no suitable device at the desired geographical location or, other times, the device has not collected/stored the data-point that is desired. However, oftentimes the missing data-points can be estimated with a very high degree of accuracy. For instance, if an application would like to know the temperature at a location where no thermometer exists, then an expert should be able to estimate the result using the values of the temperature readings in the surrounding area (for example, with a Kalman filter). Or when the application requires access

to the location of a car at some time t_1, but only the locations at time t_0 and t_2 are known ($t_0 < t_1 < t_2$), then a user versed in Newton's laws of motion should be able to calculate the midpoint-speed t_1. But how can these estimations take place in an automated fashion, without the need for human intervention?

5. **Incomplete or Inaccurate Metadata**—The solution to many of the challenges above likely lies on the extensive use of metadata. However, since much of this metadata must be manually entered by a human operator at installation time, in a massive network this will surely result in a large amount of incomplete/inaccurate information due to human error. In addition, some of this information includes characteristics that change over time (e.g., calibration parameters). Therefore, even discounting human error, the state of the metadata in the network is bound to degrade until it no longer represents reality. In these scenarios, how can missing metadata be recovered? And how can existing information be monitored and updated when necessary?
6. **Conflict Resolution**—Conflict resolution is an issue that arises mainly with actuators, but not so much with sensors. Conflicts arise, for instance, when multiple applications attempt to actuate the same device in opposing ways, or when they would like to exert mutually-incompatible changes on the environment. For example, in a scenario where a smart building is able to adapt to people's personal temperature preferences, one person may want a choose a temperature of $17C$ while the other $25C$. A human mediator would likely resolve this conflict using the average of the two temperatures, $21C$. However, a much tougher example presents itself in the actuation of pan-tilt-zoom cameras: if one application requires the camera to turn left, and the other requires it to turn right, how can the network satisfy both applications — or at least gracefully degrade their quality of service?

3 Related Work

Most middleware solutions for the IoT adopt a service-oriented design in order to support a network topology that is both unknown and dynamic. But while some projects focus on abstracting the *devices* in the network as services (such as in HYDRA[1–3], SENSEI [4], SOCRADES[5], and COBIS[6]), other projects devote more attention to data/information abstractions and their integrations with services (among which are SOFIA[1] [7], SATware[8], and Global Sensor Networks GSN [9]). A common thread throughout all of these solutions, however, is that they handle the challenge of **unknown topology** through the use of discovery methods that are largely based on the *traditional* service/resource discovery approaches of the existing Internet, ubiquitous environments and Wireless Sensor & Actuator Networks [10–12]. For instance, SOCRADES provides discovery on two levels, the device level and the service level, which can employ either standard WS-discovery (for WS Web Service) or a RESTful discovery mechanism (for RESTful services). COBIS, on the other hand, uses its own service description language COBIL[2] (Collaborative Business Item Language) where service functions and keywords are annotated with a verbal description.

[1] `http://www.sofia-project.eu`
[2] `http://www.cobil-online.de/cobil/v1.1/`

Another point of agreement in the state-of-the-art in IoT middleware is in the widespread use of semantics and metadata to overcome **heterogeneity** challenges. Indeed it is standard practice to use ontologies to model sensors, their domains, and sensor data repositories [2, 13, 14]. Some projects even go a step further and also include context information[4], or service descriptions[5]. And as a type of service composition, many projects support the concept of virtual/semantic sensors (for instance, in HYDRA, GSN and SATware), i.e. entities that abstract several aggregated physical devices under a single service. A different implementation of a similar idea, though, is provided in the SATware project: in their work, virtual sensors actually correspond to transformations applied to a set of raw sensor streams to produce another semantically meaningful stream. Although it can be said that the concept of virtual sensors is a sort of service composition, one must be careful to point out that this composition is *not fully dynamic*, in that the services are first specified at *design time*, and only then are they dynamically mapped onto the network at run time. In contrast, a much more flexible type of composition is to perform both operations at run time, through the help of small predefined composition building blocks as supported by the SENSEI and SOCRADES projects.

Regarding **scalability**, most IoT projects address this challenge by pursuing modifications in the underlying network topology. At times, this is done by adopting fully-distributed infrastructures (such as in COBIS and SOFIA), and at other times through an architecture of peer-to-peer clusters (e.g., GSN). In our view, however, while these approaches work well for the existing Internet (where traffic is made up of a relatively small amount of service interactions) they are not fit for the complex weave of interactions that will be commonplace in the Internet of Things. In the IoT, a large number of requests will involve intricate coordination among thousands of *things* and services, whereas on today's Internet most requests are largely point-to-point. Therefore, the number of packets transmitted in the network will grow strongly nonlinearly as the number of available services increases. In such an environment, performing even a simple service discovery or composition may exceed acceptable time, processing, and memory constraints. For this reason, in Section 4, we propose to address the challenge of scalability by modifying the discovery and composition algorithms themselves, rather than focusing solely on designing optimal network topologies.

Finally, among the aforementioned projects, to the best of our knowledge, *none* considers the challenges of **data-point availability**, **inaccurate metadata**, and **conflict resolution**. To address such issues, in our proposed work, we rely on the highly-structured nature of physical information. We design our middleware to support not only semantic models but also *estimation models* that perform all of these tasks transparently, in the background, without ever burdening the application with the internal details of this process. In some ways, it can be said that this aspect of our approach bears some similarity with Google's Prediction API[3]. This is a web service which allows application writers to train and use classifiers on their own datasets without requiring any knowledge of machine learning or data mining. In our work, however, we do not aim to compete with the Prediction API, but rather provide the means by which these kinds of prediction services can be leveraged without the user or application-writer even knowing about it. The process, we claim, should be realized in a completely

[3] `http://code.google.com/apis/predict/`

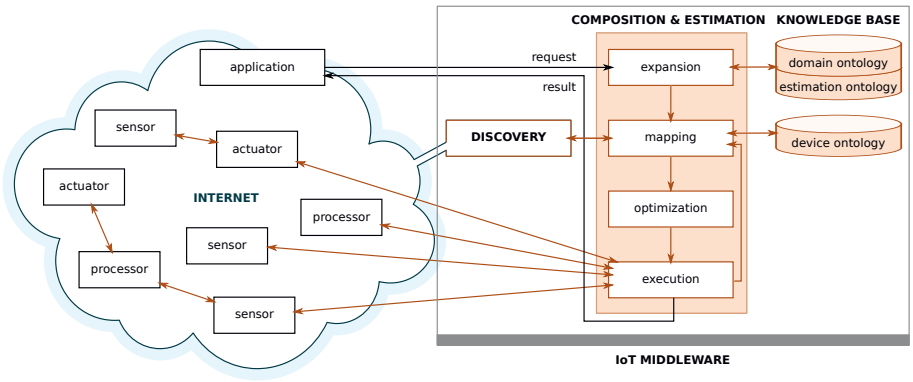

Fig. 1. Architecture of our IoT middleware

automated manner. To support all of these features, we envision an IoT architecture that is described in the next section.

4 Blueprints of a Solution

To address the challenges of the IoT, we take a service-oriented approach as commonly done in the literature, relying heavily on semantics to describe devices, their data, and their physical attributes. Building upon this semantic, service-oriented foundation, however, we also introduce the support for mathematical models capable of estimation, calibration, and conflict resolution. As a result, our proposed architecture produces a loose coupling of *things* and traditional services, all of which can contribute to resolving sensing/actuation requests, but none of which is strictly required. When one of these components is not available in the network, our middleware, then, makes use of estimations, approximations, and predictions.

Our envisioned IoT middleware, as shown in Figure 1, consists of three parts: a Discovery module, and Estimation & Composition module, and a Knowledge Base (KB). A description of these three modules, and the main innovations that we introduce in each of them, follows.

4.1 Discovery Module

The process of service discovery consists of two parts: registration and look-up. *Registration* is the process where each service connects to a server (called a registry) to give some information about itself, including a network address. *Look-up* can be described as finding the services that match a given set of desirable attributes (sensing modality, geographic location, error characteristics, etc.). However, to address the challenges of scale described in Section 2-1 we here introduce the concept of **probabilistic discovery** to provide, instead, *the set of services that can best approximate* the result that is being sought after. In the context of the example from Section 2-1, where a person would like to find out the average temperature in Paris, the middleware should proceed by first fetching the definition of “average” from the Knowledge Base, which includes a description of the well-known equation for the sampling distribution of the mean. This

equation states that in a network of M sensors, we can afford to instead use only N sensors ($N < M$) to calculate the average temperature within some mean error of e. Then, with this information the middleware should perform the following actions:

1. Use the provided error equation to estimate the number N of sensors that will be needed for this request.
2. Produce a random sample of N points in time, space, and other dimensions (such as sensor/actuator orientation in space, their coverage area, or any other attribute of a device).
3. Discover N devices in the network that approximately match those N points.
4. Given this set of devices, recalculate the error estimate.
5. Repeat 2–5, depending on whether the new error estimate is satisfactory.

The steps above are an instance of what we call a *probabilistic lookup*. That is, an intelligently-constructed query that makes use of probabilities to find approximate sets of services when exact solutions would be too costly to compute. In a similar manner, another aspect of discovery that can also be made probabilistic is the registration process. In *probabilistic registration*, services use non-deterministic functions to determine (1) which registry they should register with, (2) at what times registration should take place, and (3) what attributes they should register with each registry. Part of our upcoming research will consist on characterizing the different combinations of probability distributions that can be used throughout the discovery process (during both lookup and registration), in order to establish the advantages and disadvantages of each discovery scheme.

4.2 Composition and Estimation Module

As can be seen in Figure 1, the component that is directly in charge of processing sensing and actuation requests is the Composition & Estimation module. Within this module, the process responsible for coordinating all tasks is the composition process. "Composition" consists of finding a dataflow graph that, given a description of the input parameters and the format of the desired output, connects the available services in order to produce the desired output from the parameters.

To clarify what this means, let us consider a brute-force implementation of the composition process, consisting of the following phases:

1. *Expansion*: This step expands the initial query by replacing each term with an equivalent expression, found by traversing the domain ontology. In this brute-force implementation, the final result of the expansion phase is a set of all possible combinations of service dataflows that answer the initial query.
2. *Mapping*: This step takes all dataflows produced by the expansion step and maps them to the actual network topology. As such, mapping is necessarily performed by interacting closely with the Discovery module. This phase also interacts with our device ontology (present in the KB) that models real world devices, to complement any information found to be missing during the discovery process. The output of the brute-force mapping step is the set of all possible mappings of the input dataflows onto the network topology.

3. *Optimal mapping selection*: Once all feasible dataflows have been mapped (in the mapping phase), the IoT middleware must choose one dataflow to enact. In this phase, therefore, a dataflow that is (in some predefined way) optimal is found and passed on to the execution block, below.
4. *Execution*: Now that the best composition of services has been determined for the query, in the execution step the services are actually accessed and the result is returned (or stored). In addition, during execution, the middleware must check for any conflicts that may arise at run-time, in a process that we call conflict resolution.

In an ultra-large-scale network with a large-scale Knowledge Base, seeking an optimal composition becomes an intractable problem. So instead of pursuing an exact solution as was done in the brute-force case above, in our research we will pursue the idea of **approximately-optimal composition**, where the concepts of *expansion* and *mapping* are modified as follows:

1. *Smart expansion*: To avoid exhaustively calculating all possible equivalent sets of dataflows, only one of which will eventually be selected during the optimization phase, a much smarter approach to expansion is to instead produce a reduced set of good candidate dataflows. These candidates are the dataflows that have the highest likelihood of having a matching overlay in the network that satisfies a set of predefined constraints. An example constraint could be that the predicted execution time should fall within a certain acceptable interval.
2. *Probabilistic mapping*: Taking as input the set of candidate dataflows from the previous phase, the probabilistic mapping phase differs from regular mapping in that it does not attempt to find all possible mappings of the input dataflows into the network topology. Instead, this phase will randomly pick a small subset of all implementable mappings by making small, atomic queries to the probabilistic Discovery module. The result is a much reduced set of dataflow mappings that are computed in considerably less time and using (hopefully) orders of magnitudes less resources.

So far, all of the new features mentioned above focus on addressing the way the challenges of scale, heterogeneity, and unknown topology magnify one another. However, in addition to these challenges, we also address the challenge of unknown data-point availability (Section 2-4), by injecting in the IoT middleware enough knowledge about sensing, actuation, physics, devices, etc., to be able perform the **automated estimation** of any missing data-points. This is possible through the use of physical/statistical models that are provided *a priori* by field experts and made available in a Knowledge Base. Then, when a missing data-point is detected, or when more accurate data is requested by an application, the middleware can apply the provided models onto the timeseries of measurements from a set of sensors, and therefore estimate the most likely true value of the data at the desired spatiotemporal point. This process takes place in three steps:

1. *Model discovery*: Look in the KB for models related to the desired devices.
2. *Optimal model selection*: Pick the most appropriate models based on a few parameters and a cost function (also specified in the ontology).
3. *Estimation execution*: Apply the models to the existing historical data from sensors, using as input parameters the sensor and deployment metadata. This will be done using pre-developed engines for each model.

Furthermore, it is likely that the same framework used for automated estimation can be extended to support *auto-calibration* and *conflict-resolution* techniques, given that all of them rely on enacting expert-provided models stored in the KB. This would address the challenges described in Sections 2-5 and 2-6. In the case of the former challenge, the same physical and probabilistic models employed during *estimation* could even be reused with little or no changes: here, however, the models would be executed *backwards*, to extract the parameters given the data (i.e. calibration), rather than estimate the data given the parameters (estimation).

4.3 Knowledge Base

A key piece that is fundamental to all others listed above is a comprehensive set of ontologies describing sensors, actuators, physical concepts, physical units, etc., as well as spatiotemporal and statistical correlation models of the data. This set of ontologies goes by the name "Knowledge Base", consisting of three parts:

The *Domain Ontology* carries information about how different physical concepts are related to one another. For instance, the "wind chill" concept can be described as a function two other concepts: "temperature" and "wind speed". This ontology also links each physical concept with a set of physical units ("km/h", "m/s", etc.) as well as with the known sensors/actuators that can measure/change them.

The *Device Ontology* stores information regarding actual hardware devices that may exist in the network, including manufacturers, models, type of device, etc., and connects each device to related concepts in other ontologies (for example, which physical units it uses when outputting data, what is the device's transfer function, etc.).

Finally, the *Estimation Ontology* contains information about different estimation models ("linear interpolation", "Kalman filter", "naïve Bayesian learning", etc.), the equations that drive them, the services that implement them, and so on.

Turning once again to the example from Section 2-3, what should be clear is that even the simplest-looking requests, upon closer inspection, end up requiring a rather complex composition of services in order to obtain an accurate result. However, it is likely that once the IoT infrastructure is in place, the most common requests will certainly be *much more complex* than the examples in this paper. Therefore, one of the greatest challenges in building a middleware for the IoT lies in envisioning an architecture that can grow and adapt to new, unforeseen situations. We believe that the system outline presented here fulfills this requirement. By deriving its composition and estimation decisions from models entered by domain experts in a Knowledge Base, our middleware is built from the ground up to evolve with the ever-changing demands of future IoT applications.

5 Conclusion

We have described the core challenges of the Internet of Things, and analysed the state-of-the-art within the context of these challenges. We, then, proposed an IoT middleware that addresses these challenges through probabilities and approximations. Our middleware adopts a service-oriented architecture to abstract all sensors and actuators as services in order to hide their heterogeneity, and relies heavily on a knowledge base that

carries information about sensors, actuators, manufacturers, physical concepts, physical units, data models, error models, etc. To address challenges stemming from the IoT's massive scale and deep heterogeneity, we concentrate on three core research contributions: probabilistic discovery, approximately-optimal composition, and automated estimation. Together, these three contributions will allow our middleware to respond to sensing or actuation requests while managing the complex relationship between accuracy and time, memory, processing, and energy constraints.

References

1. Zhang, W., Hansen, K.: An evaluation of the NSGA-II and MOCell genetic algorithms for self-management planning in a pervasive service middleware. In: 14th IEEE International Conference on Engineering of Complex Computer Systems, pp. 192–201 (2009)
2. Eisenhauer, M., Rosengren, P., Antolin, P.: Hydra: A development platform for integrating wireless devices and sensors into ambient intelligence systems. The Internet of Things, 367–373 (2010)
3. Zhang, W., Hansen, K.: Semantic web based self-management for a pervasive service middleware. In: Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems, pp. 245–254 (2008)
4. Presser, M., Barnaghi, P., Eurich, M., Villalonga, C.: The SENSEI project: Integrating the physical world with the digital world of the network of the future. IEEE Communications Magazine 47(4), 1–4 (2009)
5. Guinard, D., Trifa, V., Karnouskos, S., Spiess, P., Savio, D.: Interacting with the SOA-Based internet of things: Discovery, query, selection, and on-demand provisioning of Web Services. IEEE Transactions on Services Computing 3(3), 223–235 (2010)
6. CoBIs project, Cobis final project report deliverable 104 v 2.0, Tech. Rep. (2007), `http://www.cobis-online.de/files/Deliverable_D104V2.pdf`
7. Honkola, J., Laine, H., Brown, R., Tyrkko, O.: Smart-M3 information sharing platform. In: IEEE Symposium on Computers and Communications (ISCC), pp. 1041–1046 (2010)
8. Massaguer, D., Hore, B., Diallo, M., Mehrotra, S., Venkatasubramanian, N.: Middleware for pervasive spaces: Balancing privacy and utility. In: Bacon, J.M., Cooper, B.F. (eds.) Middleware 2009. LNCS, vol. 5896, pp. 247–267. Springer, Heidelberg (2009)
9. Aberer, K., Hauswirth, M., Salehi, A.: Infrastructure for data processing in large-scale interconnected sensor networks. In: International Conference on Mobile Data Management, pp. 198–205 (2007)
10. Al-Masri, E., Mahmoud, Q.H.: Investigating web services on the world wide web. In: Proceeding of the 17th International Conference on World Wide Web, pp. 795–804. ACM, New York (2008)
11. Zhu, F., Mutka, M., Ni, L.: Service discovery in pervasive computing environments. IEEE Pervasive Computing 4(4), 81–90 (2005)
12. Meshkova, E., Riihijarvi, J., Petrova, M., Mahonen, P.: A survey on resource discovery mechanisms, peer-to-peer and service discovery frameworks. Computer Networks 52(11), 2097–2128 (2008)
13. Eid, M., Liscano, R., El Saddik, A.: A universal ontology for sensor networks data. In: IEEE International Conference on Computational Intelligence for Measurement Systems and Applications, pp. 59–62 (2007)
14. Liu, J., Zhao, F.: Towards semantic services for sensor-rich information systems. In: 2nd International Conference on Broadband Networks, pp. 967–974 (2005)