Service-Oriented Middleware for the Mobile Internet of Things: A Scalable Solution

Sara Hachem, Animesh Pathak, Valérie Issarny

▶ **To cite this version:**

Sara Hachem, Animesh Pathak, Valérie Issarny. Service-Oriented Middleware for the Mobile Internet of Things: A Scalable Solution. IEEE GLOBECOM: Global Communications Conference (Accepted), Dec 2014, Austin, United States. hal-01057530

HAL Id: hal-01057530
https://inria.hal.science/hal-01057530

Submitted on 23 Aug 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Service-Oriented Middleware for the Mobile Internet of Things: A Scalable Solution

Sara Hachem, Animesh Pathak, Valerie Issarny
Inria Paris-Rocquencourt, France
{sara.hachem, animesh.pathak, valerie.issarny}@inria.fr

***Abstract*—The Internet of Things (IoT) is characterized by a wide penetration in the regular user's life through an increasing number of mobile Things, such as mobile phones hosting sensors and actuators. However, the shift to the mobile IoT does not come without challenges, as many already existing issues remain unresolved and are amplified by the IoT scale and the mobility of its Things. The most challenging issues are handling the abundance of users and Things, providing interoperability across the heterogeneous Things, and overcoming the unknown dynamic environment due to the mobility of Things. This paper addresses the above challenges as we revisit the commonly used Service-Oriented Architecture (SOA). This leads to the design, implementation and evaluation of MobIoT, a new service-oriented middleware. MobIoT modifies standard SOA functionalities, namely service discovery, composition and access, to better address the challenges posed by the IoT, especially its scale. Specifically, MobIoT adopts probabilistic methods to decrease the number of involved devices, while building on semantic knowledge to support interoperability and fulfill users' queries for Thing-based measurements/actions.**

I. Introduction

The Internet of today is shifting towards a larger and smarter Internet known as the Future Internet [13], [15] with an essential component being the Internet of Things (IoT). The IoT is characterized by a large number of Things (e.g., phones hosting sensors and actuators, sensor-equipped vehicles, etc.), involved in every aspect of our lives, that cooperate in order to provide, among others, knowledge about the real world.

The IoT is defined as a global network infrastructure, linking physical and virtual objects through the exploitation of data capture and communication capabilities[1]. A considerable portion of those objects, is Things endowed with the ability to change their location either autonomously or, for instance, with human involvement (e.g., mobile phones, vehicles, etc.). Those mobile Things are no longer a vision for the future and they are here, within everyone's reach. For instance, all mobile phones nowadays host at least two sensors, a camera and a microphone. As of 2011, there are 5.3 billion phones users of whom more than 1 billion own a smartphone[2] with other sensors such as gyroscopes and barometers. Another example is the increasing integration of sensors and actuators in vehicles (cars with speedometers, parking sensors, etc.).

This work has been partly 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).

[1]RFIDGlobal: www.rfidglobal.eu.

[2]US Strategy Analytics: www.strategyanalytics.com.

With this wide adoption of mobile technologies, we focus our work on the mobile portion of the IoT as it introduces several benefits [17]. Firstly, with mobile devices hosting an increasing number of sensors and actuators, there is no need to spend large amounts of money on static sensor deployments. Secondly, mobile sensors can cover/sense more areas than their static counterpart. Finally, unlike static sensors, which, in many cases, are placed in remote areas, mobile Things such as mobile phones or cars are regularly recharged. It should also be noted that, given that static Things can be considered as a special class of mobile objects that are not moving, our work can be easily extended to support the IoT as a whole.

Consequently, many research efforts were directed towards building a scalable reliable mobile IoT [8], [9], [14]. Yet, as we identified in earlier work [21], many challenges remain unresolved, among which:

- *Unknown topology*: The (mobile) IoT is characterized by a network topology that is unknown and highly dynamic, due to the mobility of Things or their short life span. As a consequence, services required by an IoT application may suddenly become unavailable, because the host ran out of battery or just changed its location abruptly.
- *Heterogeneity*: The (mobile) IoT comprises sensors and actuators that are highly heterogeneous with different operating characteristics (e.g., operating platforms, sampling rates and error distributions) and different hardware characteristics (e.g., sensor chip type), hosted on diverse Things (e.g., mobile phones, vehicles, clothing, etc.) integrating many of those components. As a result, ensuring interoperability between all the Things cannot be considered as a straightforward task.
- *Scale*: To accurately represent the real world, a sensing/actuating task will most often require the cooperation and coordination of numerous (mobile) Things (within an Internet of billions). For instance, even a single application, such as calculating daily temperature variations around the globe, can require the use of millions of mobile Things resulting in an amount of information that will grow unmanageable. This is problematic due to time, memory, processing, and energy constraints.

An adequate solution towards addressing the heterogeneity and the unknown network topology issues is through a middleware that adopts a Service-Oriented Architecture (SOA) [16]. SOA is commonly used in IoT solutions [8], [9], [13], [19], to abstract Things or their measurements as services. The service-

oriented paradigm decouples the functionalities of Things from their hardware information or other technical details. Resulting services have functional attributes such as the type of measurements they provide (e.g., temperature measurement service), and non-functional attributes to specify Quality of Service information (e.g., measurement accuracy).

Traditional SOA involves three main actors that interact directly with one another (shown in Figure 1): a *service provider* (the Thing hosting the service), a *service consumer* (any IoT application), and a *Registry* for services. Moreover, any service-oriented middleware adopting this architecture supports three core functionalities: *Discovery*, *Composition* of, and *Access* to services (Figure 1). Specifically, *Discovery* is used to publish (register) services in registries that hold service metadata and to look for services that can satisfy a sensing/actuating request. *Composition* of services is used when discovered services are unable to fulfill the request. In such case, other existing services are combined to provide a new or more convenient functionality. The composed services can further be used for more complex compositions. Finally, *Access* enables the interaction with the discovered services.

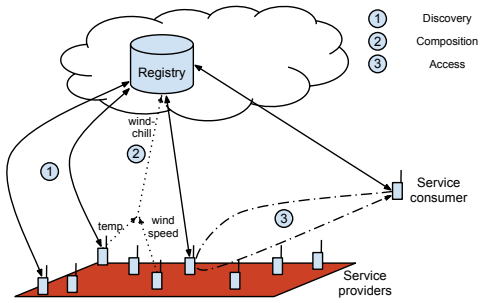

Fig. 1: The interactions in the Mobile Internet of Things.

Typically, in SOA, even if millions of services are registered, there is no need to select and access them all simultaneously. However, in the IoT, discovery, composition and access are undoubtedly more complicated. In fact, it is unlikely for a single or even a few services to be sufficient when providing real-world measurements. In most cases, to accurately represent a real-world feature, a large number of services are selected to provide their measurements, and subsequently, all acquired values should be properly aggregated. As a consequence, discovery will return a large set of accessible services, many of which can provide redundant functionalities. Consumers are then expected to access the numerous providers to acquire their measurements, over which they should know the exact aggregation/fusion logic to apply. Furthermore, such logic requires precise knowledge and understanding of the real world and its governing physics and mathematics laws. Clearly, performing discovery, composition and access tasks as presented above incurs high communication and computation costs and is thus not realistic within the large scale IoT.

In light of the above issues, our first contribution lies in *revisiting the SOA interaction patterns* to support better scalability and exempt consumers from the burden of interacting with providers. Specifically, we introduce a **Thing-based SOA** that wraps all cumbersome tasks internally in a middleware that, unlike traditional service-oriented middleware, is aware of the real world, its physics and its mathematics rules. To be inline with the new architecture and its objectives, functionalities of the middleware itself are also to be revised. To that end, we present our second contribution, **MobIoT**, a service-oriented middleware with a novel discovery protocol, which builds on a previous work on probabilistic service registration published in [10] and a novel composition protocol. MobIoT decouples the sensing/actuating tasks from the querying for measurements and requests for actions. The query should at least contain the concept to measure and the location of interest. An example of such queries would be "What is the noise level at the Colosseum in Rome?". It is important to mention that we focus on real-time discrete request/response sensing/actuation scenarios, with special interest in environment monitoring.

We proceed in the following sections to detail the proposed Thing-Based SOA and MobIoT. We start in Section II by surveying the literature for existing IoT middleware solutions. In Section III we describe the Thing-based SOA architecture, followed by Section IV where the MobIoT components are presented. We then present the evaluation of MobIoT in Section V. Finally, Section VI concludes the paper with a summary of our contributions.

II. Background

There have been extensive efforts in the literature to provide middleware solutions to realize the IoT vision. A common approach in many of those solutions is to adopt the service-oriented paradigm to support a network topology that is both unknown and dynamic, and to decouple the physical aspect of the IoT (i.e., Thing level) from its functional aspect (i.e., Thing-based service level) thus enabling better interoperability. The latter is achieved by abstracting the physical *Thing* in the network as a service (such as in HYDRA [8], [25], SENSEI [19] and SOCRADES [9]), or by abstracting the virtual Thing as a service (e.g. GSN [1]). With the former abstraction, consumers have access to the data/action sources directly while the latter provides consumers with access to processed data thus decoupling between the Thing itself and the data it generates. However, as elaborated below, there is no common approach to addressing together all the challenges we identified, and none of the solutions regards the scale issue as the central point that aggravates all other challenges.

Firstly, the **unknown topology** of the environment is addressed through discovery methods that are largely based on the *traditional* service/resource discovery approaches of ubiquitous environments [8], [9], [14]. By traditional discovery, we mean discovery of **all** appropriate devices that are reachable. For instance, authors in [14] provide a DHT-based discovery technique that accelerates the search process. Authors in [6] focus on a geographical-location based discovery where nodes publish their location, then federations are created by a managing node based on neighboring nodes. While these solutions are appropriate for small to medium scale networks, adopting traditional discovery and access within the large scale mobile IoT, involving billions of Things, is inadequate and thus, those techniques must be revisited.

Secondly, to address **heterogeneity** challenges, it is standard practice to use ontologies to model devices and their meta-

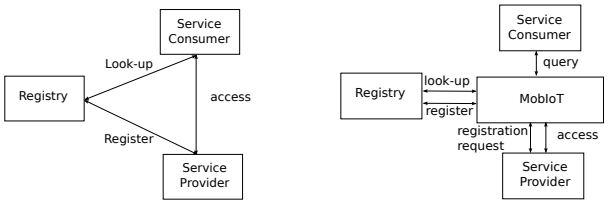

Fig. 2: a) Traditional SOA b) Thing-based SOA.

data [7], [18], context information [3], [19], or services [7], [8], [25]. Authors in [6] further model location of nodes as they consider it a very important parameter when searching for information in the IoT. Some solutions introduce the concept of virtual/semantic sensors [8], [18], [25], i.e., entities that abstract several aggregated Things under a single service as a form of composition specification. However, the semantic models they provide either model sensors, their data, their services, etc., or they model sensors and their relation to the real world (i.e., the features they measure). Yet, not much effort was directed towards providing connected ontologies that present this information on top of knowledge that goes beyond concepts measured by sensors, to include physics laws, mathematics, etc., which are at the core of the IoT.

Last but not least, regarding **scalability**, most IoT-specific solutions address this challenge by revisiting the underlying network topology (DHT-based discovery [14], geographically distributed nodes federation [6], etc.). This process can indeed render data routing, discovery, or communication more efficient. However, in our view, such modifications are not fit for the complex weave of interactions in the IoT, as the number of simultaneously active devices requesting/providing services remains too high. Consequently, a large number of requests will involve intricate coordination among millions of Things and services. In such environment, performing even a simple service discovery or composition may exceed acceptable time, communication costs and resource consumption.

To overcome the above challenges, we revisit the SOA itself along with the functionalities of traditional service-oriented middleware. Specifically, we present a Thing-based SOA that relieves service consumers from heavy communications and computations. The architecture is concretized by MobIoT, a middleware that amends conventional discovery and composition techniques to support larger numbers of Things, and be fully aware of the real world and its laws.

III. Thing-based SOA

SOA is defined as a logical way of designing a software system to provide services via published and discoverable interfaces [16]. In SOA, the provider, the registry and the consumer interact directly as shown in Figure 2(a). The provider hosts a software module that is the service implementation and publishes the descriptions (metadata) of its hosted service to the registry through which services are made discoverable. The consumer finds and selects a service description matching its requirements from the registry, and accesses the provider hosting the matching service. The interactions, involving registration, look-up and access, have the following properties:

- During registration, *all* willing service providers are able to register (publish) the descriptions of their services.
- During look-up, *all* registered services with attributes that match a request are looked-up and can be accessed.
- Aggregations/compositions are applied by consumer(s).
- Access to services is done *directly* by consumer(s).

The application of SOA to the mobile IoT results in some apparent contradictions. On the one hand, all tasks in SOA revolve around a business logic that can be satisfied by one or several services. On the other hand, in mobile IoT, all tasks and interactions revolve around what we refer to as a *Thing-based query*. The latter is a request sent by a consumer for real-world measurement/actuation tasks with the following entities:

1) *Physical concept*: real world feature to measure/actuate.
2) *Unit*: each measured concept should have a unit, otherwise the result will not be meaningful.
3) *Location*: the coordinates or name of the location of the concept to measure. It can be a point in space or an area.

With such queries, it is unlikely to have only one or just a few services that can provide accurate answers to represent a real-world feature. Hence, expecting the service consumer to interact with the numerous service providers individually to access their services and acquire their measurements, then know how to treat each and every value (with different possible formats, types, units, etc.), in addition to the aggregation logic to apply, requires high communication and computation capabilities that the consumer will most likely not possess. As an alternative, our approach [11], depicted in Figure 2(b) revisits the SOA and renders most interactions and heavy computations transparent to the consumer that is only expected to know the sought after measurement, as follows:

- *Probabilistic Registration*: The registration of a provider's service is probabilistic. The goal is to allow only a subset of willing providers to register their services depending on how well registered ones can substitute it. Precisely, the decision is based on whether or not the mobility path to be followed by the new Thing will be covered by other mobile Things, i.e., whether or not other registered Things with similar sensing/actuating services will be present at its future locations when it crosses them.
- *Probabilistic Look-up*: The look-up is also probabilistic and it returns only a subset of services based on the total area coverage they can provide. We adopt the same logic as in intrusion detection solutions [23], [24] where the spatial distribution of sensors has a major effect on the performance of the sensing system. Based on those solutions, when measuring a feature over some *area* (e.g., temperature in Rome), we sample sensors from a Uniform distribution in space so that sensors from all over that area have the same likelihood of being selected. However, when the concept of interest is at a *specific point* in space, a better distribution would be Normal as it selects more sensors around that point and less as we move farther.
- *Thing-based Composition*: All aggregations/compositions are executed transparently by the middleware based on accurate physics and mathematics knowledge encoded in a supporting ontology. The middleware identifies alternative concepts (real-world features) that should be provided as input to a mathematical formula whose output

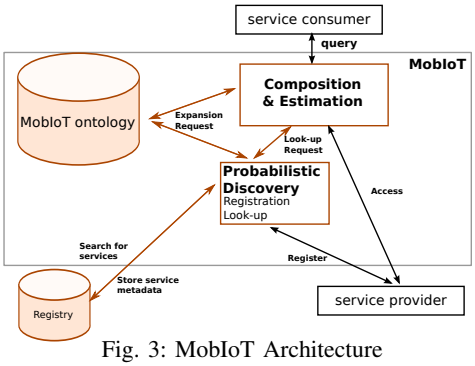

Fig. 3: MobIoT Architecture

is an estimate of the value of the concept of interest. It is of utmost importance when either no Thing in the network hosts services that can directly measure the required concept, or when more accuracy and therefore more data sources are required.

IV. MobIoT Design

MobIoT was designed to transparently provide the functionalities required by the proposed Thing-based SOA. The core components of the middleware are depicted in Figure 3.

A. IoT ontology

A key piece to our middleware is a comprehensive ontology, describing sensors, actuators, physical concepts, etc., as well as spatio-temporal and statistical correlation models of their data. We build our ontology on top of a set of NASA's SWEET ontologies (http://sweet.jpl.nasa.gov/ontology/.), used to model real world information, mathematics and physics, etc. For details on our ontology we refer the reader to [12].

B. Discovery

Discovery is the component that wraps probabilistic *Registration* and *Look-up* functionalities as follows:

- The Registration component generates the decision to allow or prevent the new Thing from registering its services. The component estimates whether or not the path of the new Thing can be covered by other mobile Things. To that end, the component computes the probability that any of them be present at each of the locations on the path of the new Thing, after which it compares the resulting probability value to a required sensing coverage. We consider that the coverage requirement (threshold) depends on the sensor and can be specified in our ontology. Only if the resulting probability is lower than the threshold, can the Thing register its service. We use the **Truncated Lévy Walk** mobility model [20] to estimate the mobility of registered Things and compute the probabilities above. The component can use any other mobility model as long as the corresponding mathematical formulas to compute the probabilities are provided. As shown in [10], our registration solution successfully limits the registration of redundant services.
- The Look-up component is in charge of returning a subset of services to access that can satisfy the Thing-based query. Based on the requested measurement and the location of interest, the component determines the most adequate probability distribution and the number of needed services. This number is computed based on a percentage of the area of interest to be sensed/acted on by the selected subset. The result is then forwarded to the registry to determine the actual Things to sample.

C. Composition & Estimation

The service composition in MobIoT consists of three steps:

1) *Expansion*: This step recursively replaces each concept in the user query with a set of equivalent concepts that together can provide an estimate of the value of the initial requested concept. The estimate is computed by mathematical formulas modeled in our ontology.
2) *Mapping*: This step takes the set of expansion concepts as input and determines, through the Look-up component, the potential providers that can measure those concepts and the addresses of their hosted services to be able to access them. The output of the mapping step is the set of service addresses.
3) *Execution*: In the execution step, the services are actually accessed and the individual results are returned. All measurements over the same concept are aggregated based on an appropriate *fusion function* defined in our ontology for each concept. Results are then passed as parameters to the mathematical functions found by the expansion step, and the final result is returned to the user. This phase also handles the actual **access** to services.

D. Registry

The Registry is not a simple storage service that holds the metadata and descriptions of services. In fact, it is designed to assist the probabilistic look-up component as it performs the heavy computations and holds information on the locations of Things when they register their services. Using this information, the registry can estimate their location at the time of look-up requests and know which of them best fits in the required spatial probability distribution.

V. MobIoT Implementation & Evaluation

MobIoT is implemented as a library using Java 1.6 and deployed on Android phones and personal computers. The Registry is implemented as a RESTful Web service. For the ontology, we use Jena (http://jena.apache.org/) for laptop deployments and Androjena (http://code.google.com/p/androjena/) for Android deployments. The code was in particular used as part of the CHOReOS European project on choreographies for the Future Internet (http://www.choreos.eu) and assessed by project partners through use case studies [4], [5], more details can be found in [11]. Our code is available as open source at http://forge.ow2.org/projects/choreos/.

A. Experimental Setup

Our objective for the evaluations below is twofold: assess MobIoT's usability, and scalability. For the former, we created a proof-of-concept base noise sensing application deployed on Samsung Galaxy S3 phones, meant to illustrate the benefits of

exploiting MobIoT when integrating sensing/actuation services in mobile applications. For scalability evaluations, we simulate concurrent requests for noise measurements at the same location for 100 to 1,000 requests per second. To simulate mobile Things, we use a synthetic mobility trace generated with SUMO [2], which generates traces (15,000) based on real data in the city of Cologne [22]. We executed the simulation in a cluster of 40 machines with Scientific Linux 5.5, Intel Xeon X5650 dual processors and 48 GB RAM.

B. MobIoT Usability

The mobile application we created can be used to: i) register noise sensing services hosted on mobile devices; and ii) acquire noise level information at a location of interest. It comprises two components: i) a GUI allowing users to choose between registering their sensing services at their current location or acquiring noise estimations at some location; and ii) MobIoT, through which the composition, discovery and access to mobile Things take place. While writing the application code, the developer is only required to call `execRegQuery()` to request a registration decision for the base noise sensing service (Listing 1) and `getData()` to request noise data (Listing 2). Consequently, MobIoT alleviates a large portion of programming efforts by wrapping the logic specific to the IoT domain.

```
boolean tryToRegister(long locationLong, long locationLat){
                return execRegQuery(ThingID,''getnoise'',''
                    noise_service'',''noise_sensor'',''
                    noise'',''http://arles.rocq.inria.fr
                    :8080/1234/getnoise',locationLong,
                    locationLat,myPath,''dB'');}
```

Listing 1: Code snippet for `execRegQuery()` method call.

To execute the `tryToRegister()` method, the only information required is the coordinates of the registering Thing. The developer is required to provide, in the `execRegQuery()` method, the Thing ID, service name, sensor name and concept to measure, in addition to the accessible address of the service, the future mobility path of the Thing and the noise unit (dB).

```
SensorData getNoise(String locName, long latitude, long
    longitude) {
        Query myQr = new Query(new Selector(new Concept("
            noiselevel")), new Constraint(new Where("
            noiselevel.unit.equals=dB")), new Location(
            locName, latitude, longitude));
        return getData(myQr, Constants.DOUBLENOISE);}});}
```

Listing 2: Code snippet for `getData()` method call.

To execute the `getNoise()` method, the only information required is the name and coordinates of the location of interest. The developer should specify that the Query is over noise (the concept), with a constraint being that the unit of measurement should be in dB and the location of interest being the location name and coordinates passed as parameters to the `getNoise()` method. Afterwards, the created query and the desired datatype should be passed to the `getData()` method.

C. MobIoT Scalability Evaluation Results

In order to assess the scalability and illustrate the benefits introduced by MobIoT, we conducted a set of experiments that

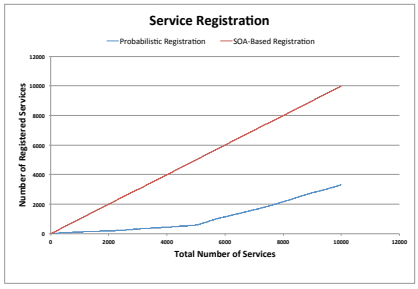

Fig. 4: A comparison of the number of registered services following the SOA and Thing-based SOA approaches.

can be divided into two categories: i) the first set compares MobIoT's probabilistic registration, in terms of the number of registered services to SOA-based registration, and ii) the second set compares MobIoT's probabilistic look-up and access to standard SOA, look-up and access. The performances in the latter are evaluated with respect to one metric: the Query Response Time (QRT). The QRT for registration is the time taken for MobIoT to compute the registration decision and if allowed, register the Thing's service. The QRT for look-up and access is the time needed for MobIoT to execute a user query, including finding and accessing services (with no composition). It should be noted that if composition was needed, it takes place transparently and automatically without involving developers in the process.

a) Registration scalability.: We evaluated the benefits of employing the probabilistic registration by comparing the number of registered services following standard SOA to the number of registered services following our Thing-based SOA. As illustrated in Figure 4, while all 10000 Things register sequentially in the deterministic approach, our introduced approach leads to at least 60% less Things registering their services and consequently less communication and computation costs, especially upon lookup and access. The number of successfully registered services is presented on the Y-axis, while the total number of services to register is presented on the X-axis. Note that at the beginning of the registration evaluation, the first 5000 Things that attempt to register are located in each other's vicinity, explaining the low registration rate illustrated by the probabilistic registration curve between 0 and 5000 services on the X-axis. Afterwards, the Things start to show up in more sparse areas, which leads to a higher registration rate illustrated by the portion of the probabilistic registration curve between 5000 and 10000 services on the X-axis.

b) Look-up and access scalability.: We also measured the response times for concurrent queries with both standard SOA registration, look-up and access and probabilistic registration, look-up and access. The benefits of our probabilistic look-up are clearly illustrated in Figure 5 depicting the distribution of response times for both approaches with 1,000 concurrent user queries. Figure 5(a) shows that most occurrences of QRT for the SOA approach are between 0 and 10 seconds while Figure 5(b) shows that most occurrences of QRTs for the probabilistic approach are between 0 and 3 seconds. On the one hand, in the SOA approach, MobIoT retrieves all registered services at each query and attempts to access them all in order to provide a result. On the other hand,

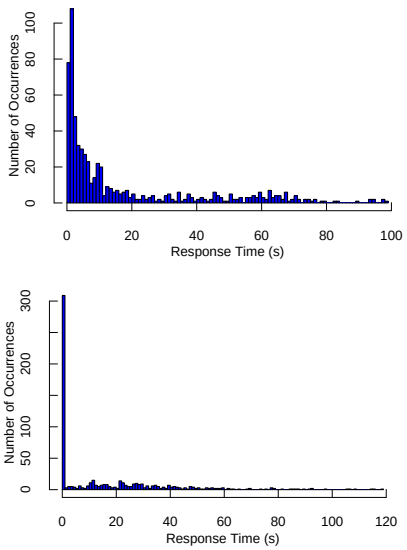

Fig. 5: The QRT time distribution for $1,000$ queries with a) SOA registration, look-up b) probabilistic registration, look-up.

for the probabilistic approach, response times are lower as the system retrieves a small subset of registered services to access.

In conclusion, the experiments performed in this section illustrate the usability and scalability of MobIoT with the easy-to-implement methods provided by the middleware and the strongly decreased registration rate and response times when answering a user query as compared to standard SOA discovery and access.

VI. Conclusion

We presented in this paper a Thing-based SOA and a middleware designed to address the large scale, heterogeneity and unknown environment issues of the mobile IoT. To that end, the Thing-based SOA revisits traditional SOA to wrap cumbersome tasks in a smart middleware, MobIoT, rendering them transparent to the consumer. Unlike existing SOA solutions, MobIoT exploits knowledge of the real world, and its mathematics and physics laws. It relies on semantic technologies to address the heterogeneous nature of Things, in addition to specifying compositions. We address the large scale issue through a novel twofold probabilistic discovery approach that controls the participation of Things in a sensing/actuation task. Firstly, the service registration is probabilistic. It builds on the fact that paths of mobile Things in dense networks are bound to cross and can substitute one another based on the services they host. Secondly, look-up is probabilistic, and only a subset of registered services is selected based on their hosts' distribution in space and the type of the event. Our approach also allows the substitution of a service by a composition of the functionality of other types of registered services. We implemented our middleware and evaluated its performance to illustrated its introduced benefits, assess its usability and demonstrate its scalability. Additionally, as part of our future work, we plan on comparing the performance of MobIoT to that of existing SOA middleware solutions.

References

[1] 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)

[2] Behrisch, M., Bieker, L., Erdmann, J., Krajzewicz, D.: SUMO - Simulation of Urban Mobility: An Overview. In: The Third International Conference on Advances in System Simulation, (SIMUL). Spain (2011)

[3] Chen, Y., Chen, Y.: Context-Oriented Data Acquisition and Integration Platform for Internet of Things. In: Technologies and Applications of Artificial Intelligence (TAAI), Conference on. pp. 103–108. IEEE (2012)

[4] CHOReOS consortium: DynaRoute Architectural Design (D8.2). Tech. rep. (2011), http://www.choreos.eu/bin/Download/Deliverable

[5] CHOReOS consortium: Passenger Friendly Airport Services Choreographies Design (D6.2). Tech. rep. (2011), http://www.choreos.eu/bin/Download/Deliverable

[6] Christophe, B.: Managing massive data of the Internet of Things through cooperative semantic nodes. In: Semantic Computing (ICSC), 2012 IEEE Sixth International Conference on. pp. 93–100. IEEE (2012)

[7] De, S., Barnaghi, P., Bauer, M., Meissner, S.: Service modelling for the Internet of Things. In: Computer Science and Information Systems (FedCSIS), Federated Conference on. pp. 949–955. IEEE (2011)

[8] Eisenhauer, M., Rosengren, P., Antolin, P.: HYDRA: A Development Platform for Integrating Wireless Devices and Sensors into Ambient Intelligence Systems. The Internet of Things pp. 367–373 (2010)

[9] 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)

[10] Hachem, S., Pathak, A., Issarny, V.: Probabilistic registration for large-scale mobile participatory sensing. In: Pervasive Computing (2013)

[11] Hachem, S.: Service-Oriented middleware for the large-scale mobile Internet of Things. Ph.D. thesis, Université de Versailles-Saint Quentin en Yvelines (2014)

[12] Hachem, S., Teixeira, T., Issarny, V.: Ontologies for the Internet of Things. Proceedings of the PhD Student Symposium of the ACM/IFIP/USENIX 12th International Middleware Conference (2011)

[13] Issarny, V., Georgantas, N., Hachem, S., Zarras, A., Vassiliadist, P., Autili, M., Gerosa, M., Hamida, A.: Service-oriented middleware for the Future Internet: state of the art and research directions. Journal of Internet Services and Applications 2(1), 23–45 (2011)

[14] Paganelli, F., Parlanti, D.: A DHT-based discovery service for the Internet of Things. Journal of Computer Networks and Communications (2012)

[15] Papazoglou, M., Pohl, K., Parkin, M., Metzger, A.: Service research challenges and solutions for the Future Internet: S-cube-towards engineering, managing and adapting service-based systems. Springer (2010)

[16] Papazoglou, M.: Service-oriented computing: concepts, characteristics and directions. In: Web Information Systems Engineering. Proceedings of the Fourth International Conference on. pp. 3 – 12 (2003)

[17] Pereral, C., Zaslavsky, A., Christen, P., Salehi, A., Georgakopoulos, D.: Capturing sensor data from mobile phones using global sensor network middleware. In: Personal Indoor and Mobile Radio Communications (PIMRC), 23rd International Symposium on. pp. 24–29. IEEE (2012)

[18] Pfisterer, D., Romer, K., Bimschas, D., Kleine, O., Mietz, R., Truong, C., Hasemann, H., Kroller, A., Pagel, M., Hauswirth, M., et al.: SPITFIRE: toward a semantic Web of Things. Communications Magazine, IEEE pp. 40–48 (2011)

[19] 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. Communications Magazine, IEEE pp. 1–4 (2009)

[20] Rhee, I., Shin, M., Hong, S., Lee, K., Kim, S., Chong, S.: On the levy-walk nature of human mobility. IEEE/ACM Transactions on Networking (TON) 19(3), 630–643 (2011)

[21] Teixeira, T., Hachem, S., Issarny, V., Georgantas, N.: Service Oriented Middleware for the Internet of Things: A Perspective. In: Towards a Service-Based Internet, LNCS, vol. 6994, pp. 220–229. Springer Berlin Heidelberg (2011)

[22] Uppoor, S., Trullols-Cruces, O., Fiore, M., Barcelo-Ordinas, J.M.: Generation and analysis of a large-scale urban vehicular mobility dataset. IEEE Transactions on Mobile Computing 99(PrePrints), 1 (2013)

[23] Wang, Y., Fu, W., Agrawal, D.: Gaussian versus uniform distribution for intrusion detection in wireless sensor networks. Parallel and Distributed Systems, IEEE Transactions on PP(99) (2012)

[24] Yaqin, W., Xiaoliang, X.: Sensor deployment distribution effect on intrusion distance in wireless sensor networks. Innovations in Bio-inspired Computing and Applications, International Conference on pp. 103–106 (2011)

[25] 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)