A Semantic Engine for Internet of Things: Cloud, Mobile Devices and Gateways

Amelie Gyrard, Soumya Kanti Datta, Christian Bonnet
Mobile Communication
Eurecom
Biot, France
Email: {gyrard, dattas, bonnet}@eurecom.fr

Karima Boudaoud
Rainbow team
Laboratoire I3S-CNRS/UNSA
Biot, France
Email: karima@polytech.unice.fr

***Abstract*—Semantics is becoming a requirement in Internet of Things (IoT). Recent works underline future challenges for IoT: (1) integrate semantic web technologies, (2) provide interoperability, (3) interpret IoT data, (4) ease the development of IoT applications, and (5) fit to the different requirements of people and applications. In this paper, we design a semantic engine to meet these requirements and integrate it in different components available in IoT architectures. To the best of our knowledge, the main novelty of this paper is the semantic engine flexible enough for different IoT architectures where semantics is integrated in: (1) cloud, (2) devices such as smart phones or tablets, and (3) M2M gateways. As proof of concepts, we discuss deployment of the proposed semantic engine at cloud systems and mobile devices. Moreover, we show that such deployments are coherent with ETSI M2M and oneM2M standardizations.**

Keywords—Internet of Things (IoT), Machine-to-Machine (M2M), Semantic Web of Things (SWoT), Architectures, Applications.

I. Introduction

Miorandi et al. [1] clearly explain a lack of standardization related to models and data formats and describe the need for: (1) cross-domain applications, and (2) semantic interoperability and data management for exchanging and analyzing IoT/M2M data to infer useful information and to ensure interoperability among IoT applications and for reasoning. Ozpinar explains that resolving the meaning of data is a challenging problem and without processing it, the data is useless [2]. He also discusses the challenging problems regarding heterogeneity of billions of devices and he outlines that the challenge of resource-constrained devices has to be taken into account. In 2014, Chen et al. introduce the need for intelligent processing for IoT data and explain the issue related to domain specific-applications: applications cannot combine the data from different silos [3].

We analyzed recent works and we highlight the following requirements in future semantic-based IoT architectures:

- Requirement A: Provide interoperability among applications, even from heterogeneous domains.
- Requirement B: Interpret IoT data.
- Requirement C: Ease the development of IoT applications.
- Requirement D: Fit to the different needs of people (e.g., constrained devices, ensure privacy).

The semantic engine addresses such requirements. The main novelty of this work is to show that the semantic engine can be integrated in different components of IoT architectures: cloud, mobile devices and gateways to fulfill the requirements mentioned above of different applications and the needs of various existing projects.

The rest of the paper is structured as follows. Section II presents the state of the art and clearly explains the limitations. Section III describes semantic-based IoT architectures for cloud, mobile devices and gateways. Section IV is focused on proof of concepts. Section V demonstrates the applicability of the semantic engine on standardizations such as ETSI M2M and oneM2M and on a distributed architecture. Finally, we conclude the paper in section VI.

II. State of the Art

In this section, we present existing works related to: (1) integrating semantics in IoT architectures, (2) integrating semantics in gateways, (3) integrating semantics in embedded devices, and (4) semantic-based distributed architecture. Finally, we highlight the limitations of the existing works.

A. Semantic-based IoT

Bassi et al. design the IoT-A architecture reference model (ARM) [4]. They are not focused on the interpretation of data and do not ease the development of interoperable applications. Kiljander et al. design a semantic-based architecture for Pervasive Computing and Internet of Things [5]. They underline the need for a common way to abstract the heterogeneity of devices. In their architecture, agents share semantic information with each other. They do not employ semantic to enrich IoT data. Their architecture is based on the IoT-A ARM.

B. Semantic-based Gateways

Bonino et al. design the Domotic OSGi Gateway (Dog) Gateway, an ontology-powered middleware based on the OSGi framework and the DogOnt ontology to support the integration of different networks and support logic-based intelligence [6]. Dog is an open-source solution capable of running on low cost hardware such as Rasperry Pi. This work is focused on smart buildings and has not be applied to other domains. Further, the reasoning part is not explained in detail. The dog gateway has been deployed in two real use cases.

Desai et al. design the Semantic Gateway as Service (SGS), a bridge between sensors and end-user applications [7]. They integrate the W3C SSN ontology, the SemSOS tool and domain ontologies in their gateway to semantically annotate sensor data. They explicitly describe that for domain specific applications, the gateway can be equipped with additional domain specific ontologies. They do not underline the difficulties to reuse domain ontologies relevant for IoT and interoperability issues to interpret heterogeneous sensor data descriptions.

C. Semantics in embedded devices

Hasemann et al. design the Wiselib TupleStore to store RDF data in embedded devices [8]. The database is stored in either the flash memory or RAM. It runs on different platforms specific to sensors such as Contiki and TinyOs operating systems or more popular operating system for mobile phones such as Android and iOS.

D. Semantic-based distributed architecture

The following works use semantics for distributed reasoning or storage but they have not been applied in the context of IoT.

In 2005, the Distributed Reasoning Architecture for a Galaxy of Ontology (DRAGO) distributed reasoning system, implemented as a peer-to-peer architecture, is designed by Serafini et al. [9]. The goal of DRAGO is to reason on distributed ontologies. Kaonp2p has been designed by Haase et al. to query over distributed ontologies [10]. LarKC (Large Knowledge Collider) is a scalable distributed platform [11]. Marvin is a scalable platform for parallel and distributing reasoning on RDF data [12]. Schlicht et al. propose a peer-to-peer reasoning for interlinking ontologies [13]. In 2012, Abiteboul et al. see the Web as a distributed knowledge base and propose an automated reasoning over this knowledge base [14].

In 2013, WebPIE (Web-scale Parallel Inference Engine) is an inference engine for semantic web reasoning (OWL and RDFS) based on the Hadoop platform designed by Urbani et al [15]. WebPIE is scalable over 100 billion triples [16]. Coppens et al. propose an extension to the SPARQL query language to support distributed and remote reasoning. For their implementation, they extend the Jena ARQ query engine [17]. In 2014, Park et al. propose a semantic reasoning based on their XOntology and SPARQL. They use the Hadoop platform, HDFS and MapReduce to deal with thousands of sensor data nodes [18].

Hartig et al. discover datasets which could be relevant for a specific SPARQL query to interact with the Web of Data [19]. Khadilkar et al. propose Jena-HBase, a distributed, scalable and efficient triple store [20]. Since, the triple store provided by Jena is TDB, a single-machine RDF storage, they propose to combine the Jena triple store with the Hadoop framework, more precisely, the HBase distributed database, the distributed files system HDFS to store data, MapReduce for processing data stored in HDFS. Husain et al. propose a scalable, distributed and highly fault tolerant framework to handle billions of RDF triples, they describe a scheme to store RDF data in HDFS (Hadoop Distributed File System) and use Hadoops MapReduce framework to answer the queries [21] [22]. Kulkarni et al. propose a distributed SPARQL query engine using MapReduce [23]. Quilitz et al. propose DARQ, a query engine for federated SPARQL queries. The user executes one SPARQL query which queries several SPARQL endpoints [24] .

1) Limitations of these works: We scrutinized the related works and highlighted the following limitations:

- Semantic-based distributed architectures have not been applied to the context of Internet of Things.
- Existing semantic-based IoT architectures are not enough flexible for cloud, mobile devices and gateways. Further, they do not fulfill all of the requirements mentioned in the introduction.
- Few works propose concrete approaches to combine domains.
- Most of the semantic-based architectures are not focused on the processing of data to infer high-level abstractions.
- In addition to these limitations, semantic web technologies are not considered in some IoT architecture [25] [26].

III. Semantic-Based IoT Architectures

We designed a semantic-based IoT architecture flexible enough to support the requirements mentioned in the introduction.

The main novelty of the semantic engine (see Figure 1) is that it can be adapted inside different architecture components: 1) cloud, (2) devices, and (3) gateways. To build IoT applications on constrained devices, there is a need to filter and get only what we need to build the application. For this reason, our semantic engine has been adapted to constrained devices, by filtering only a subset of the components that are required to build the applications. Our semantic engine is composed of five main semantic-based components, as displayed in Figure 1 and Figure 2:

- **Template Catalogue** is a dataset of pre-defined semantic-based IoT templates with pre-selected interoperable domain ontologies, datasets and rules required to interpret IoT data. Each template is adapted to a specific sensor type and the domain where the sensor is deployed. The knowledge (ontologies, rules and datasets) associated to this template enables to semantically annotate IoT data, infer high-level abstractions and even provide suggestions. This component is essential to meet: (1) Requirement A to provide interoperability among applications, (2) Requirement B to interpret IoT data, and (3) Requirement C to ease the development of IoT applications.
- **Converter** that semantically annotates data. Since data comes from heterogeneous and common projects, there is a necessity to describe data in an interoperable manner. An unified language to describe interoperable IoT data has been implemented in the converter [27]. This language is a basis to later interpret IoT data. This component is essential to meet: (1) Requirement

B to interpret IoT data, and (2) Requirement C to ease the development of IoT.

- **Reasoning engine** to get high-level abstractions from IoT data. This component enriches and interprets data with background knowledge and is essential to meet Requirement B to interpret IoT data.
- **Query engine** to query smarter data used in the final applications. This component is essential to meet Requirement C to ease the development of IoT.
- **Final application** that interacts with end-users. This component is essential to meet Requirement D to fit to the different needs of people (e.g., constrained devices, ensure privacy).

Finally, when the last component has been executed, the application gets suggestions and parse them to display them in a user-friendly interface, send notification or even control actuators.

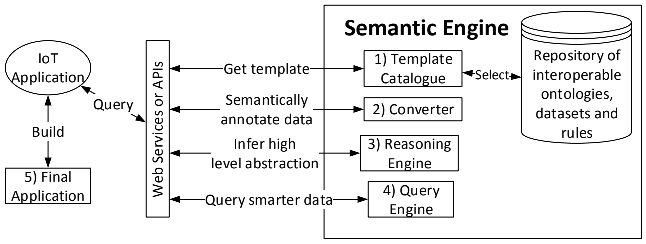

Fig. 1. Semantic engine for Internet of Things

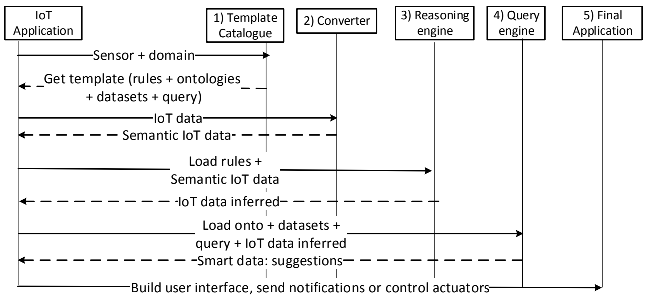
IoT Application
1) Template Catalogue
2) Converter
3) Reasoning engine
4) Query engine
5) Final Application
Sensor + domain
Get template (rules + ontologies + datasets + query)
IoT data
Semantic IoT data
Load rules + Semantic IoT data
IoT data inferred
Load onto + datasets + query + IoT data inferred
Smart data: suggestions
Build user interface, send notifications or control actuators

Fig. 2. Semantic engine operational flow in IoT applications

A. *Semantic engine embedded on Cloud*

We designed a semantic and cloud-based IoT architecture as displayed in Figure 3. Firstly, the application looks for a semantic-based IoT template to develop the future application by interacting with the **Template Catalogue**. According to the two parameters (sensors and domains) indicated, the application downloads the template required to build the application. Secondly, the files available in the template allow the application to semantically annotate sensor data with the **Converter**. Thirdly, the application loads the rules provided in the template and the semantic IoT data and executes the **Reasoning engine** component to get high level abstractions. Then, **Query engine** component is executed with the SPARQL query provided in the template to get suggestions. Finally, according to the high level abstractions or suggestions returned, the user-friendly interface can be designed, or send notifications or even control actuators in the **Final Application**.

The main advantage of having the semantic engine in the cloud-based architecture is to send data to the cloud and then interpret data without considering performance issues. This approach is relevant when data do not need to be secured or when privacy is not required. Further, we assume there is always a connection to the Internet to send data and get back inferred data. One of the benefits of the cloud-based architecture is the scalability to run the reasoning and query engines with 'Big Data'.

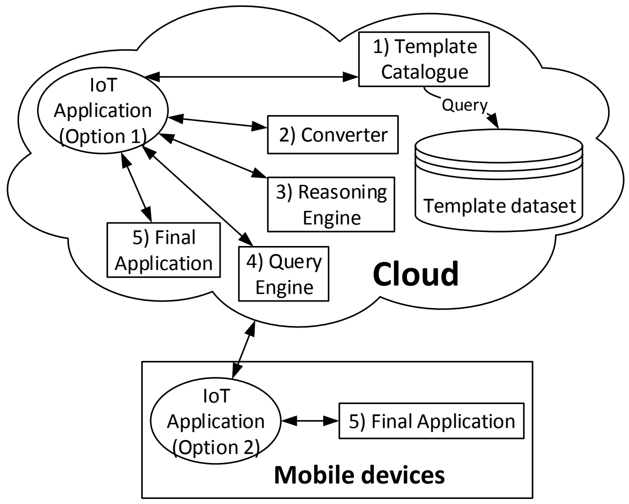

Fig. 3. Semantic engine in cloud-based architecture

B. *Semantic engine embedded on Mobile Devices*

The semantic engine is integrated in mobile devices as displayed in Figure 4. In this architecture, a first connection to the Internet is required to download the template by querying the **Template Catalogue**. Then, the components **Converter**, **Reasoning and Query Engine** and **Final Application** are precessed locally and Internet connectionless.

The main advantage of this architecture is to avoid sending sensor data to the cloud and process the data locally. Information abstraction can be processed locally to reduce the deluge of data on network communications. It brings two main advantages: network traffic reduction and the enhancement of comprehensiveness for the end-user. To conclude, communications costs, latency, and privacy issues when sending data to the cloud are avoided.

C. *Semantic engine embedded on Gateways*

The semantic engine is integrated in gateways as displayed in Figure 5. Smart gateways can expose, exchange and integrate data in ways unforeseen at design time. The main advantage is to avoid to send data to the cloud and to process the data locally. Users' data is kept where it belongs and does not have to be centrally stored. To conclude, communications costs, latency, and privacy issues when sending data to the cloud are avoided.

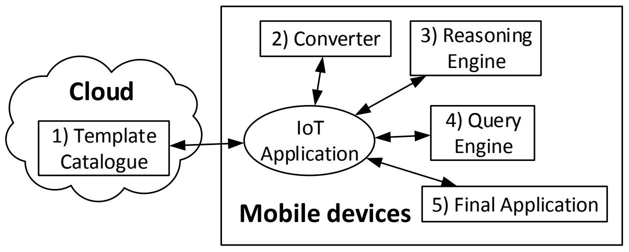

Fig. 4. Semantic engine in mobile device based architecture

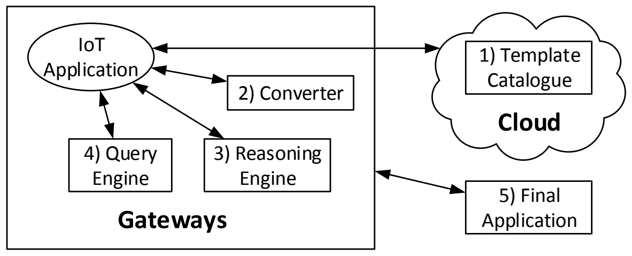

Fig. 5. Semantic engine in gateway based architecture

D. Discussions

We propose several deployment scenarios depending on the size of the problem to solve. If there is only 1 sensor and 1 domain involved, it is possible to embed the semantic engine on mobile devices. In case of smart cities scenarios, the semantic engine would be better on the cloud. In case of personal healthcare, the semantic engine would be better on the mobile device to ensure privacy.

In Figure 6, we sum up advantages and drawbacks of each architecture.

Architecture	Pros	Cons
Cloud-Cloud or Cloud-Device	• Large computation power • Large Scalability	• Communication cost • Access to Internet • Security/Privacy issues
Device	• Privacy ensured • No Communication Cost	• Small computation power • Low Scalability
Gateway	• Privacy ensured • No Communication Cost	• Medium computation power • Medium Scalability

Fig. 6. Advantages and drawbacks of each architecture

IV. Implementation

In this section, we explain that the semantic engine has been implemented as proof-of concepts in two architectures presented above: cloud and mobile devices.

A. Cloud-based architecture

The cloud-based architecture is available on the web[1]. The proof-of concept has been implemented with Java 1.7 and Google Application Engine. The Jena framework [28] has been used to build semantic web applications. Jena enables loading ontologies, execute reasoning engines and query engines. The template is loaded with the Jena framework, the reasoning engine with the Jena reasoning, and the query engine with the Jena ARQ engine. Jena is also used in the converter to semantically annotate IoT data. The final application has been implemented with HTML, CSS, Javascript, AJAX and Bootstrap. Ajax gets the results returned by web services. The different components are available on the cloud and can be used through user-friendly interface or APIs. We designed RESTful web services: (1) to query templates, (2) for pre-defined scenarios, and (3) for the converter, etc. The documentation is available online[2].

B. Mobile device-based architecture

We implemented our approach for Android, which is an increasingly popular operating system not only for smart phones or tablets but also for many other future devices such as TVs. The mobile device-based architecture has been implemented on Android-powered devices such as mobile phones and tablets. For technical reasons, we used the AndroJena library instead of Jena, to run the semantic processing on Android devices. The AndroJena framework has been used to interpret IoT data and build the Semantic Web of Things application.

V. Semantic Engine Integrated in Standardizations

In this section, we explain how the different components of the semantic engine are integrated in architectures designed by standardizations such as ETSI M2M [29] and oneM2M [30]. Moreover, we explain that we could design a semantic-based distributed architecture to enable smart devices exchanging high-level abstractions with each other.

A. Semantic-based ETSI M2M architecture

In this section, we design a semantic-based M2M architecture [31] inspired by the ETSI M2M architecture. The main goals are to: (1) get sensor measurements from heterogeneous domains, (2) semantically annotate and interpret M2M data, (2) combine domains with each others to build cross-domain M2M applications. In our proposed semantic-based

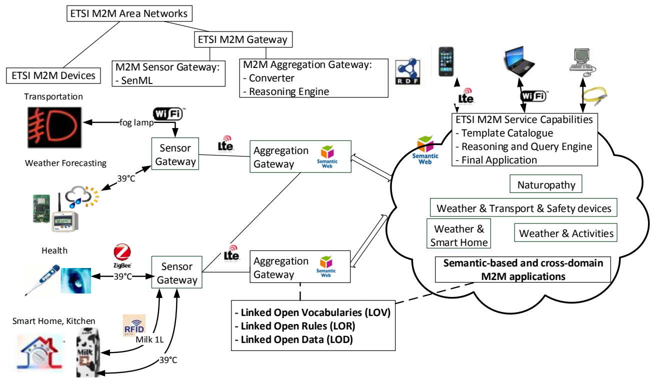

Fig. 7. Semantic engine embedded in our semantic-based ETSI M2M architecture

[1]http://sensormeasurement.appspot.com/

[2]http://www.sensormeasurement.appspot.com/?p=documentation

M2M-based architecture depicted in Figure 7, we integrate semantic web technologies both in M2M gateways and M2M applications. We propose two kinds of M2M gateways due to various treatments: (1) M2M sensor gateways, and (2) M2M aggregation gateways.

The M2M sensor gateways retrieve M2M measurements provided by heterogeneous M2M devices and include the acquisition interface to support heterogeneous protocols such as RFID, Bluetooth, 6LowPan, CoAP and Zigbee. Several formats can be used such as SenML or SWE to get sensor metadata. We use the lightweight SenML protocol to retrieve heterogeneous sensor measurements for a first and quick implementation. SenML provides simple sensor measurements: the name, the value, the unit and the date (e.g., temperature 5 DegC). SenML or SWE bridges the gap of interoperability of heterogeneous sensor data but does not provide descriptions such as 'this temperature is a body temperature' or 'the milk is produced by cows and contains lactose'. For these reasons, we propose to enrich M2M data with semantic web technologies. The sensor gateways forward the SenML data to the aggregation gateways.

The M2M aggregation gateways semantically annotate sensor metadata with the **Converter** component based on semantic web languages (RDF, RDFS, OWL). The M2M aggregation gateway semantically annotate SenML data to provide unified sensor measurements and add an explicit context since data comes from by heterogeneous domains and projects. This step is essential to later interpret data (**Reasoning Engine** component).

Sophisticated semantic treatments are performed in M2M applications through semantic-based reasoning and semantic web technologies such as the SPARQL language to query sensor data, the Linked Open Data, the Linked Open Vocabularies and the Linked Open Rules to enrich sensor metadata with external domain knowledge. Such treatments are done in the **Reasoning and Query Engine** component. M2M applications produce the knowledge required to semantically annotate data and interpret it thanks to the **Template Catalogue** component. the **Reasoning Engine** component performed the reasoning on heterogeneous semantic measurements. An example of **Final Application** is the naturopathy application to suggest recipes according to the mood, diets, diseases, ingredients available in the kitchen, according to the season, etc. Query such smart data is possible thanks to the **Query Engine** component. This example shows that four sensor networks need to be merged: health, smart kitchen, weather forecasting and emotion sensor networks.

B. Semantic-based oneM2M architecture

The semantic-based oneM2M architecture comprises several components [32] that we remind below by matching them to our different proposed components. As displayed in Figure 8, we integrated the different components in the semantic-based architecture. The *Semantic Annotation* integrates the **Converter** component to semantically annotate sensor data in an interoperable manner. The *Data Analytics* corresponds to the **Reasoning engine**) component by running the reasoning engine to interpret sensor data. The *M2M Applications* correspond to the **Final Application** component. The *Data Repository* to store IoT data. The *Semantic Repository* to store semantic IoT data annotated with Resource Description Framework (RDF) [33]. The *Ontology Modeling and Processing* corresponds to the **Template Catalogue** component. The *Reasoning* engine to deduce new knowledge. The *Semantic Mash-up* corresponds to our web services, user interfaces or APIs. The *Semantic Analysis and query* corresponds to our **Query Engine**.

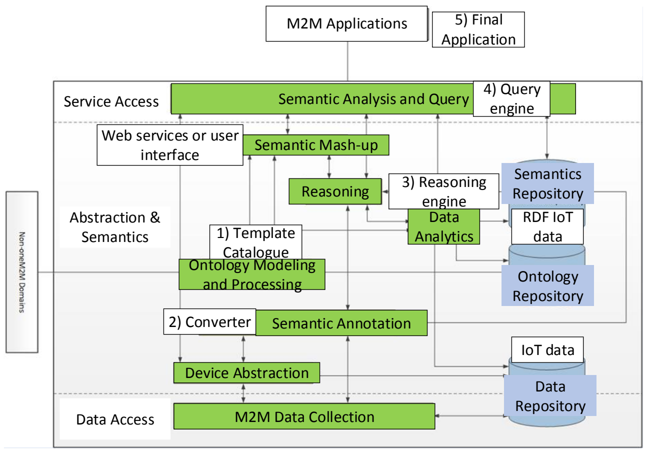

Fig. 8. Semantic engine embedded in the oneM2M architecture [32]

C. Distributed Semantic-based IoT Architecture

Distributed architecture would enable smart objects to speak with each other based on an unified language to describe sensor measurements. This unified language has been implemented in the **Converter**. Objects can interface with each other thanks to Semantic Web Services [34]. Semantic Web Service exploit the interoperable domain knowledge defined in the template catalogue.

Each device will provide Web Services and SPARQL endpoints to open access to its data, high-level abstraction and even suggestions to other devices. In Figure 9, Federated SPARQL queries enable querying distributed semantic IoT data.

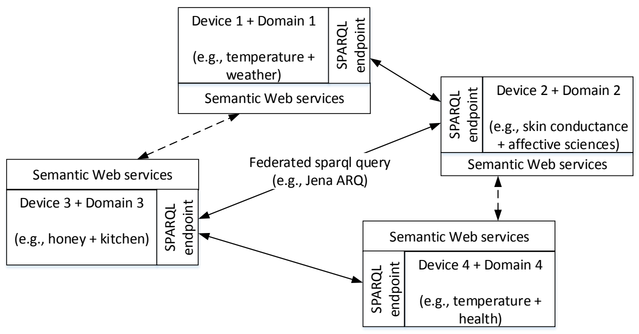

Fig. 9. Query distributed RDF sensor data

A limitation of this architecture is to handle different versions of interoperable ontologies, rules and datasets when these knowledge bases need to be updated with more background knowledge.

VI. Conclusion

To the best of our knowledge, this is the first work flexible enough to integrate a semantic engine either on the cloud, constrained devices or gateways. Moreover, we have shown that such architectures are compatible with standardizations such as ETSI M2M and oneM2M. As future work, we would like to work on the implementation of the distributed architecture to enable smart devices exchanging high-level abstractions or even suggestions with each other through semantic web services. Another future work is to provide APIs and web services for each component of the semantic engine.

Acknowledgment

The authors would like to thank colleagues and reviewers for their valuable feedback and fruitful discussions. This work is supported by the Com4Innov Platform of Pole SCS[3] and DataTweet[4] (ANR-13-INFR-0008).

References

[1] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, 2012.

[2] M. Öspinar, "A flexible semantic service composition framework for pervasive computing environments," Ph.D. dissertation, Moddle East Technical University, 2014.

[3] S. Chen, H. Xu, D. Liu, B. Hu, and H. Wang, "A vision of iot applications challenges and opportunities with china perspective," 2014.

[4] A. Bassi, M. Bauer, M. Fiedler, T. Kramp, R. Van Kranenburg, S. Lange, and S. Meissner, *Enabling things to talk*. Springer, 2013.

[5] J. Kiljander, A. D'Elia, F. Morandi, P. Hyttinen, J. Takalo-Mattila, A. Ylisaukko-Oja, J. Soininen, and T. Salmon Cinotti, "Semantic interoperability architecture for pervasive computing and internet of things," 2014.

[6] D. Bonino, F. Corno, and L. De Russis, "A semantics-rich information technology architecture for smart buildings," *Buildings*, vol. 4, no. 4, pp. 880–910, 2014.

[7] P. Desai, A. Sheth, and P. Anantharam, "Semantic gateway as a service architecture for iot interoperability," *arXiv preprint arXiv:1410.4977*, 2014.

[8] H. Hasemann, A. Kröller, and M. Pagel, "The wiselib tuplestore: A modular rdf database for the internet of things," *arXiv preprint arXiv:1402.7228*, 2014.

[9] L. Serafini and A. Tamilin, "Drago: Distributed reasoning architecture for the semantic web," in *The Semantic Web: Research and Applications*. Springer, 2005, pp. 361–376.

[10] P. Haase and Y. Wang, "A decentralized infrastructure for query answering over distributed ontologies," in *Proceedings of the 2007 ACM symposium on Applied computing*. ACM, 2007, pp. 1351–1356.

[11] D. Fensel, F. van Harmelen, B. Andersson, P. Brennan, H. Cunningham, E. Della Valle, F. Fischer, Z. Huang, A. Kiryakov, T.-I. Lee *et al.*, "Towards larkc: a platform for web-scale reasoning," in *Semantic Computing, 2008 IEEE International Conference on*. IEEE, 2008, pp. 524–529.

[12] E. Oren, S. Kotoulas, G. Anadiotis, R. Siebes, A. ten Teije, and F. van Harmelen, "Marvin: Distributed reasoning over large-scale semantic web data," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 7, no. 4, pp. 305–316, 2009.

[13] A. Schlicht and H. Stuckenschmidt, "Peer-to-peer reasoning for interlinked ontologies," *International Journal of Semantic Computing*, vol. 4, no. 01, pp. 27–58, 2010.

[14] S. Abiteboul, E. Antoine, and J. Stoyanovich, "Viewing the web as a distributed knowledge base," in *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*. IEEE, 2012, pp. 1–4.

[15] J. Urbani, S. Kotoulas, J. Maassen, F. Van Harmelen, and H. Bal, "Webpie: A web-scale parallel inference engine using mapreduce," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 10, pp. 59–75, 2012.

[16] J. Urbani, "On web-scale reasoning," Ph.D. dissertation, 2013.

[17] S. Coppens, M. Vander Sande, R. Verborgh, E. Mannens, and R. Van de Walle, "Reasoning over sparql," in *Proceedings of the 6th Workshop on Linked Data on the Web*, 2013.

[18] K. Park, Y. Kim, and J. Chang, "Semantic reasoning with contextual ontologies on sensor cloud environment," *International Journal of Distributed Sensor Networks*, vol. 2014, 2014.

[19] O. Hartig, C. Bizer, and J.-C. Freytag, "Executing sparql queries over the web of linked data," in *The Semantic Web-ISWC 2009*. Springer, 2009, pp. 293–309.

[20] V. Khadilkar, M. Kantarcioglu, P. Castagna, and B. Thuraisingham, "Jena-hbase: A distributed, scalable and efficient rdf triple store," Technical report, 2012. http://www. utdallas. edu/~vvk072000/Research/Jena-HBase-Ext/tech-report. pdf, Tech. Rep., 2012.

[21] M. F. Husain, P. Doshi, L. Khan, and B. Thuraisingham, "Storage and retrieval of large rdf graph using hadoop and mapreduce," in *Cloud Computing*. Springer, 2009, pp. 680–686.

[22] M. F. Husain, L. Khan, M. Kantarcioglu, and B. Thuraisingham, "Data intensive query processing for large rdf graphs using cloud computing tools," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*. IEEE, 2010, pp. 1–10.

[23] P. Kulkarni, "Distributed sparql query engine using mapreduce," *Master of Science Computer Science School of Informatics University of Edinburgh*, pp. 18–31, 2010.

[24] B. Quilitz and U. Leser, "Querying distributed rdf data sources with sparql," in *The Semantic Web: Research and Applications*. Springer, 2008, pp. 524–538.

[25] S. K. Datta, C. Bonnet, and N. Nikaein, "An iot gateway centric architecture to provide novel m2m services," in *Internet of Things (WF-IoT), 2014 IEEE World Forum on*. IEEE, 2014, pp. 514–519.

[26] S. K. Datta and C. Bonnet, "Smart M2M gateway based architecture for M2M device and Endpoint management," in *ITHINGS 2014, IEEE International Conference on Internet of Things 2014, September 1-3, 2014, Taipei, Taiwan*, Taipei, TAIWAN, PROVINCE OF CHINA, 09 2014. [Online]. Available: http://www.eurecom.fr/publication/4318

[27] A. Gyrard, S. K. Datta, C. Bonnet, and K. Boudaoud, "Standardizing generic cross-domain applications in Internet of Things," in *GLOBECOM 2014, 3rd IEEE Workshop on Telecommunication Standards: From Research to Standards, December 8, 2014, Austin, Texas, USA*, Austin, UNITED STATES, 12 2014. [Online]. Available: http://www.eurecom.fr/publication/4412

[28] B. McBride, "Jena: A semantic web toolkit," *Internet Computing, IEEE*, vol. 6, no. 6, pp. 55–59, 2002.

[29] E. M2M, "Machine-to-Machine Communications (M2M); Study on Semantic support for M2M data, ETSI Techinal Report 101 584 v2.1.1 (2013-12)," 2012.

[30] OneM2M, W. M. Abstraction, and Semantics, "oneM2M Technical Report 0007 Study of Abstraction and Semantics Enablement v.0.7.0, Study of Existing Abstraction and Semantic Capability Enablement Technologies for consideration by oneM2M," 02 2014.

[31] A. Gyrard, "A machine-to-machine architecture to merge semantic sensor measurements," in *Proceedings of the 22nd international conference on World Wide Web companion*. International World Wide Web Conferences Steering Committee, 2013, pp. 371–376.

[32] OneM2M, W. M. Abstraction, and Semantics, "oneM2M Technical Report 0007 Study of Abstraction and Semantics Enablement v.2.3.0, Study of Abstraction and Semantic Enablements," 01 2015.

[33] O. Lassila and R. R. Swick, "Resource description framework (rdf) model and syntax specification," 1999, http://www.w3.org/TR/REC-rdf-syntax/.

[34] S. A. McIlraith, T. C. Son, and H. Zeng, "Semantic web services," *IEEE intelligent systems*, vol. 16, no. 2, pp. 46–53, 2001.

[3]http://www.pole-scs.org/

[4]http://www.agence-nationale-recherche.fr/?Projet=ANR-13-INFR-0008