Performance Evaluation Suite for Semantic Publish-Subscribe Message-oriented Middlewares

Fabio Viola*, Alfredo D'Elia*, Luca Roffia† and Tullio Salmon Cinotti*†
*ARCES - University of Bologna, Bologna, Italy - 40125
†DISI - University of Bologna, Bologna, Italy - 40126
Email: {fabio.viola2, alfredo.delia4, luca.roffia, tullio.salmoncinotti}@unibo.it

***Abstract*—The emerging Internet of Things paradigm is driving the industry and the research towards Information and Communication Technologies (ICT) scenarios supporting high heterogeneity and interoperability. We claim that software architectures based on Semantic Publish-Subscribe Message-Oriented Middlewares (SPS-MoMs) are a powerful approach to address the requirements of such scenarios. While benchmarks and frameworks are available to evaluate the performance of MoMs, Semantic Web tools (i.e., SPARQL endpoints and RDF stores) and publish-subscribe systems, there are still no de-facto standards for the evaluation of SPS-MoMs, due to the novelty of this approach. In this paper, we propose Performance Evaluation Suite (PES), a benchmarking framework aimed at retrieving relevant performance indicators about a generic SPS-MoM. The feasibility of the proposed approach is proved by using PES to compare different implementations of a Semantic Information Broker (SIB), the core component of a SPS-MoM named Smart-M3.**

***Keywords*—Message-oriented middleware; benchmark; performance evaluation; semantics; IoT.**

I. INTRODUCTION

In the last decade, the Information and Communication Technologies (ICT) world has seen the birth of a new paradigm known as Internet of Things (IoT) [1]. Researchers from different areas have been involved in studying and strengthening the vision behind IoT. This new paradigm revolutionized the way the Internet worked up to ten years ago: laptops, PCs, tablets and smart phones are surrounded by (and need to communicate with) heterogeneous smart objects (i.e., things) spread in the physical environment. Smart objects continuously produce (i.e., sensors) and consume (i.e., actuators) data in order to provide services in different application domains (Asin and Gascon listed more than 50 application domains [2]) ranging from transportation [3][4] to logistics [5], from healthcare [6][7] to entertainment [8], from agriculture [9][10] to smart buildings [11][12], just to name a few.

Dealing with such heterogeneity in terms of application domains (e.g., different requirements), networks and protocols (e.g., DASH7 [13], 6LoWPAN [14], MQTT [15], COAP [16], XMPP [17], AMQP [18]) and device capabilities (e.g., power consumption [19]) ask for new interoperable and scalable solutions. We claim that the level of interoperability, dynamicity, flexibility, expressivity and extendibility required in IoT could be provided by a Message-Oriented Middleware (MoM) [20], more specifically a Semantic Publish-Subscribe MOM (SPS-MoM). On one hand, the MOM interaction paradigm allows to cope with events generated by IoT devices and the publish-subscribe mechanism provides an asynchronous and highly scalable many-to-many communication model, granting decoupling in terms of space, time and synchronization. On the other hand, the use of Semantic Web [21] technologies (i.e., Resource Description Framework (RDF) [22], Web Ontology Language (OWL) ontologies [23] and SPARQL 1.1 language [24]) is functional to achieve interoperability at information level. In fact, OWL ontologies allow the representation rich and complex knowledge about application domains in the form of RDF graphs that can be queried and updated using the SPARQL 1.1 language.

The main drawback of Semantic Web technologies concerns the low level of performance that makes it difficult to achieve responsiveness and scalability required in many IoT applications. The main reason for the poor performance is that Semantic Web technologies have been designed to process data sets consisting of big amounts of RDF triples that evolve constantly but at a much slower rate compared to the rate of elementary events occurring in the physical environment. Frameworks, benchmarks and methods for performance evaluation of Semantic Web systems, in general, and Semantic Publish-Subscribe systems, in particular, have been proposed in the literature. Unfortunately, these methods are not suitable for analyzing the performance of a Semantic Publish-Subscribe MOM. In fact, the former (e.g., [25][26][27][28]) are mainly designed to evaluate the performance of a SPARQL endpoint on answering a predefined set of queries with reference to several data sets and they do not include any SPARQL Update. The latter are instead focused on analyzing the performance of specific publish-subscribe systems (e.g., [29][30]).

In this paper we present, a suite dedicated to the evaluation of the performance of Semantic Publish-Subscribe MOMs. The implementation of this general suite was then specialized, without loss of generality, on the Smart-M3 platform [31], where publish and subscribe primitives are both expressed using SPARQL 1.1 (i.e., respectively as SPARQL Update and SPARQL Query) or through a RDF triple pattern serialization formalisn named RDF-M3. The main contribution of our work consists in a set of tools and methods to evaluate all the relevant performance metrics by executing existing benchmarks or creating user defined ones, specific to the target application domain. A benchmark definition includes the definition of the updates and queries (e.g., SPARQL) along with the definition of the RDF data set (e.g., OWL, N3). Tools are used to populate the knowledge base and to run the benchmarks. The evaluation outcome is in the form of graphical representations of the main results (i.e., SVG or PNG files) and includes the statistical analysis on the measured timing components (e.g., mean, variance, maximum and minimum values included in a CSV file). Finally, an example of the evaluation of two Smart-M3 SIBs (i.e., OSGi SIB [32], RedSIB [33]) is presented.

 ISBN: 978-1-61208-505-0

The article is organized as follows: after a review of the related work, an overview of the reference platform is reported in Section III. Then, a detailed description of the evaluation suite software architecture is presented in Section IV. The subsequent section reports on the evaluation of existing SIBs. We conclude in section VI.

II. Related Work

As stated by Guo et al. in [34], benchmarking a Semantic Web system is a challenging task. The main research questions concern the benchmark definition and the design of a suite able to run the same benchmark on different systems. The answers to these two questions become also more difficult moving from Semantic Web systems to Semantic Publish-Subscribe Message-Oriented Middlewares (SPS-MoMs). In fact, in a Semantic Web system, the aim is in general evaluating the performance of the query mechanism implemented by the underpinning SPARQL endpoint, while in a SPS-MoM the focus is more on the subscription mechanism. The latter assume that the benchmark defines not just the set of queries (i.e., that can be used as set of subscriptions), but also the set of updates and how these two sets interact (i.e., which updates trigger which subscriptions). Concerning the benchmark definition, Guo et al. proposed the Leigh University Benchmark (LUBM) [35] aimed at benchmarking Semantic Web knowledge base systems in large OWL applications. LUBM provides a knowledge base (whose ontology is called *univ-bench*) and a set of 14 queries designed to validate the knowledge base management system and its query engine. The starting knowledge is provided by the Univ-Bench Artificial Data generator (UBA), a tool generating a complete data set regarding the University domain. The correctness, response time and completeness (evaluated on explicit statements or implicit knowledge available through reasoning) are taken in consideration to provide the performance profile for the knowledge base. Considering SPS-MoMs, this benchmark allows to evaluate the time response of SPARQL queries, but it is not suitable to evaluate the subscription mechanism (i.e., it does not specify any SPARQL update).

The University of Freiburg proposed a Benchmark for SPARQL endpoints called SP^{2}B [36]. This benchmark is based on the DBLP dataset containing open bibliographic information on major computer science journals and proceedings [37]. SP^{2}B is provided with a data generator that produces an N3 file [38] containing n triples (where $n \in \{10k, 50k, 250k, 1M, 5M, 25M\}$). The query set is made up of 17 SPARQL queries (14 SELECT, 3 ASK) for which is known the exact number of results, depending on the dimension of the knowledge base. This benchmark is designed to assess the performance of the SPARQL query engine (functionality and processing speed). Another relevant benchmark for the SPARQL language is [39] but the research objective originating the benchmark definition was not to evaluate or compare SPS-MOMs, but to choose between a native Semantic architecture or one obtained through a SPARQL to SQL rewriter.

Concerning the design of a benchmarking suite for semantic publish-subscribe systems, to the best of our knowledge, [40] is the most representative work. Despite the approach being quite similar to the one here presented, some differences can be clearly appreciated. First, the update sequence is supposed to be generated pseudo-randomly, while we specify the update profile as an input. Having a predefined update set allows to better control the experiment. Second, there is not a clear distinction between the software modules and this can limit the extensibility and flexibility of the solution. Third, it is not possible to configure complex sequences of operations in order to make the performance analysis deeper. Instead our Performance Evaluation Suite (PES) is specifically designed to be modular. The user is able to configure the experiments and to obtain charts and detailed log files including important statistics such as the variance of the elapsed time, the minimum, maximum and mean value. Since the performance in SPS-MoMs are often affected by the content and size of the knowledge base, PES also allows to repeat every experiment on different data sets.

III. The Reference Platform

The Performance Evaluation Suite has been designed with a general approach and the first target platform chosen has been Smart-M3 [31]. Smart-M3 is an interoperability platform developed since 2008. This platform has been adopted in several past and ongoing research projects, like Internet of Energy [41], Arrowhead [42] and CHIRON [43] just to name a few. The development of Smart-M3 is currently carried on by several European universities and the proposed solution has been applied in different application domains like e-health, smart energy systems [44] and tourism [45][46].

The central component of the Smart-M3 platform is the Semantic Information Broker (SIB) that is aimed at storing the shared knowledge base in the form of an RDF graph. Several implementations of the SIB exist: 1) RedSIB [47] is a C general-purpose implementation, fast and nowadays very diffused; 2) the OSGi SIB [32] is a more recent work oriented at IoT gateways; 3) pySIB [48] is a lightweight Python implementation developed for low-powered computing nodes as, for example, System on Chips (SoCs) devices; 4) CuteSIB [49] is another recent implementation born as a fork of the old RedSIB.

Knowledge Processors (KPs) represent the client side of each application: they share data through the SIB and interoperate thanks to proper messages encoded with the Smart Space Access Protocol (SSAP). KPs can be developed exploiting one of the many existing APIs (currently available for Java, Python, C, C#, Ruby, Javascript, PHP).

The architecture of the Smart-M3 platform is summarized in Fig. 1.

The Smart-M3 interoperability platform allows to update and retrieve data using primitives based on SPARQL or on a formalism known in the Smart-M3 literature as RDF-M3, based on the concept of triple patterns. A triple pattern traces the model of an RDF triple, but allows the use of wild cards for the subject, the predicate and the object. If B, U and L are respectively the sets of the possible BNodes, URIs and Literals, a triple is defined as: $t = (s, p, o)$ where $s \in B \cup U$, $p \in U$, and $o \in U \cup B \cup L$. Introducing the wild card "Any" or "*" (that corresponds to a specific URI and matches every term) a triple pattern can be defined as: $t = (s, p, o)$ where $s \in B \cup U \cup \{*\}$, $p \in U \cup \{*\}$, and $o \in U \cup B \cup L \cup \{*\}$. In example a pattern

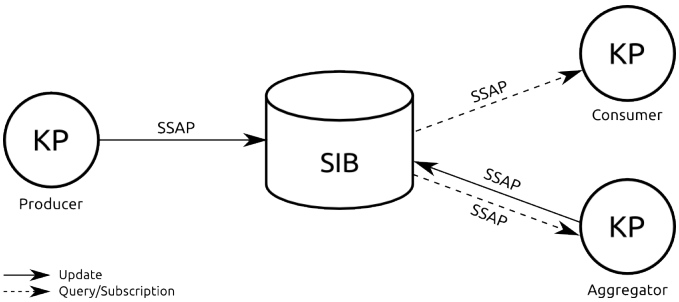

Fig. 1. The architecture of the Smart-M3 interoperability platform

based query $(*, rdf : type \in U, ns : Person \in U)$ retrieves every triple where the predicate is the URI rdf:type and the object is the URI ns:Person allowing to build a list of all the persons stored in the SIB. The SIB can be queried using a list of triple patterns: the result is made up by all the triples matching at least one of the provided triple patterns. In some cases pattern based interaction is more intuitive and simple for developers, however for inserting or retrieving complex graphs the SPARQL language [24] is always the best choice.

In Section V, the proposed PES is used to benchmark the performance of the RDF-M3 query and update mechanism of two SIBs. The results of these benchmarks are then compared with similar tests performed against the SPARQL query and update engines of the same SIBs. The results demonstrates how in some cases RDF-M3 outperforms SPARQL.

IV. Software Architecture

The PES is a free set of software modules released under the GNU General Public License 3.0. The entire suite is developed with the Python programming language and it is based on the C implementation of the Python interpreter, often referred to as CPython. PES is multiplatform, so it supports all the major operating systems. The PES software architecture is shown in Fig. 2 and described in the following subsections.

A. The Configuration Manager

The PES behavior depends on the directives specified in its configuration files (compliant with the specifications contained in [50]) and from the command line. The principal parameters specified from the command-line or through the global configuration file are the list of the SIBs to be tested (composed by IP address and port and by the required interaction protocol, e.g., SSAP [31] or JSSAP [48]) and the type of test to be performed (e.g., a query test).

Other configuration files are test-specific and are used to configure the desired benchmark. A benchmark is defined by proper configuration files. Each of these configuration files allows to specify the initial knowledge base, the number of iterations to perform, the desired output format for the chart (i.e., SVG or PNG) and if the CSV output file should be produced or not. Depending on the type of test to be performed, the configuration file may include different sections.

B. The KB Loader

The Knowledge Base Loader (KB Loader) is used to load the triples that initially constitute the knowledge base when a performance test is started. This component currently supports the N3 and the OWL KB serialization formats. The first allows to be compatible with the SP^{2}B benchmark [36], since its data generator produces an N3 file. The KB Loader sends n triples at a time to the SIB, where n is a parameter whose value depends on the trade off between KB size, number of operations to load it and efficiency of the target SIB to process large input files.

C. The PES Core

The core of PES is composed of the test modules. This extensible set of modules is currently composed of an Update Test, a Query Test and a Subscription Test.

1) Update Test: allows to measure the performance of an update request with either SPARQL or RDF-M3. For all the SIBs to be tested, the module performs a series of insertions of n triples where n ranges from n_{MIN} to n_{MAX} with step s. Each of these parameters is configured exploiting the Configuration Manager described in Subsection IV-A. Every test is repeated I times, here I is the number of iterations requested to obtain sufficient statistical samples. The mean value, the minimum and maximum and the variance are then calculated.

The time elapsed to perform the update operation is measured at the client side, so it can be considered as the sum of different components:

$$\begin{aligned} t_{update} = & t_{kp\_req} + t_{net\_req} + \\ & t_{sib\_req} + t_{sib\_elab} + t_{sib\_rep} + \\ & t_{net\_rep} + t_{kp\_rep} \end{aligned} \quad (1)$$

where t_{kp_req} and t_{kp_rep} respectively represent the time needed by the Knowledge Processor to encode the request and parse the reply, t_{net_req} and t_{net_rep} are the number of milliseconds used to transfer the packets over the network and t_{sib_req}, t_{sib_elab} and t_{sib_rep} represent the time used by the context broker to parse the received request, elaborate the request and produce a reply. The current implementation of the PES only measures t_{update}.

Measuring the time elapsed to perform an update allows to assess whether or not the SIB is able to timely store and share the information sent by the KP. The module can be configured to run with active subscriptions to evaluate their impact on the platform.

2) Query Test: The Query Test module measures the performance of the SPARQL engine (whether the requested query is a SPARQL one) or of the underlying RDF store (in case the requested query formalism is RDF-M3). For each formalism, two kinds of tests can be performed:

- *Simple test*: the knowledge base is loaded, then the query is performed;
- *Complex test*: the knowledge base is loaded in several steps and at the end of each step the specified query is performed.

The module can be configured, as in the previous case, exploiting the Configuration Manager described in Subsection

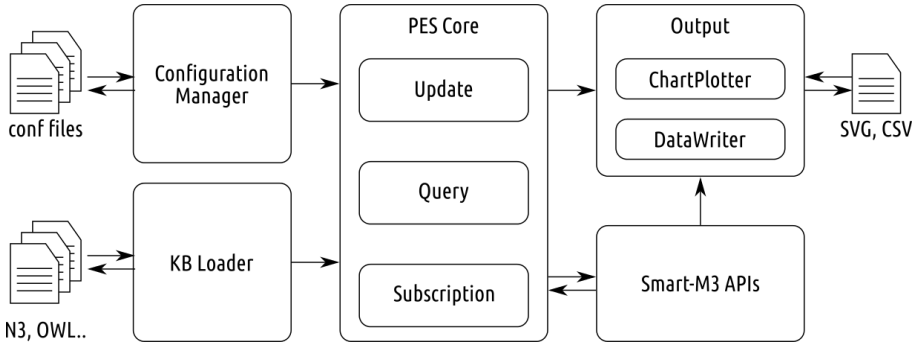

Fig. 2. The Software Architecture of the Performance Evaluation Suite

IV-A. The parameters used to set the behavior of the module are:

- The type of query test to perform (i.e., *simple* or *complex*);
- The files containing the knowledge base to load together with their format (i.e., N3 or OWL) and the desired step;
- The query to perform together with its type (i.e., SPARQL or RDF-M3);
- The number of iterations to perform.

The measured time t_{query} can be considered as the sum of different components, the same highlighted for the Update Test. For each test the minimum, the maximum and mean values of the time elapsed are returned, together with the variance. A CSV file gathers all the information deriving from the execution of a test and, if desired, an SVG chart is plotted according to the settings in the test configuration file.

3) Subscription Test: represents our most significant contribution since, to the best of our knowledge, none of the existing benchmarks allows to properly characterize the performance of a semantic publish-subscribe platform.

This test allows to subscribe to a given triple pattern using RDF-M3 or to a subgraph using the SPARQL QUERY language, then to perform updates of the knowledge base and measure the time in milliseconds required by the KP to receive the expected notification. The Subscription Test can also be used to instantiate a variable number n of KPs, each one with the same subscription, in order to calculate a notification loss ratio or to perform stress tests.

The Subscription Test can be configured with a dedicated configuration file that states the initial knowledge base (a list of n3 or OWL files to load), the subscriptions and the updates to perform and the desired number of iterations.

D. The Ouptut module

The Output module reports the results of the tests performed by plotting the related charts and writing all the measured values on a CSV file. The module relies on the pygal library that allows to render the charts on SVG or PNG files.

In Section V, it is possible to observe the charts rendered by this module, while in the following listing it is reported an example of a CSV file produced during the execution of a subscription test. The first field is the name of the SIB tested and from the second field a list of the collected notification times. The row is concluded by the mean value, minimum and the maximum values and the variance. All the values here reported are expressed in ms.

```
S0,2.819,...,2.986,1.792,3.948,0.281
S1,3.789,...,2.538,1.381,3.789,0.57
S2,1.054,...,2.392,1.003,3.51,0.673
```

E. The Smart-M3 APIs

Knowledge Processors are developed through proper APIs that make possible the interaction with the SIB. The APIs are not developed ad-hoc for the purpose of this project, but are external modules included into the PES. Since PES is developed in Python, the APIs adopted by the suite are the Python Smart-M3 APIs (including the one providing support for the JSSAP introduced by pySIB [48]).

The update mechanisms, the query functionalities and the subscription engine represents the targets of the tests modules forming the PES Core. The PES is not constrained to the Smart-M3 platform, but replacing this module with the proper APIs (and replacing the function calls to such APIs) can be used to evaluate the performance of other Semantic Publish-Subscribe MOMs.

V. EVALUATION

The configuration adopted for these benchmarks is composed of a server called `mml` and a host called `desmodue` connected in a Local Area Network at 1GBps. The former, `mml`, is a server provided with 12 core Intel Xeon CPU E5-2430 v2 at 2.50 GHz, 15.360 Kb cache and 32 GB RAM and 280 GB hard disk. The latter, `desmodue`, is a laptop PC with a CPU Intel Core(TM) i5-2520 at 2.50 GHz with 3.072 KB of cache for every core. This host is provided with 4 GB RAM and 160 GB hard disk. `desmodue` runs Linux Mint 17 Qiana, while `mml` Ubuntu Server 16.04.

PES runs on the host named `desmodue`, while the SIBs to be tested runs on `mml`. In this demonstrative tests we decided to take into account the C implementation of the SIB named RedSIB [47] and the Java one, called the OSGi SIB [32], both supporting the standard SSAP protocol and both executed without a persistent storage.

 ISBN: 978-1-61208-505-0

A. Update Test

A demonstration of the Update Test is shown in Fig. 3. The test perfomed on the two above-mentioned SIBs consists in measuring the time elapsed to insert a block composed by n triples with an RDF-M3 update; n varies from 250 to 2500 with a step of 250. No subscriptions were active during the test. Three iterations of the test have been performed. The RDF-M3 update builds the triples exploiting information coming from the configuration file that contains the desired namespace (i.e., `ns` bound to `http://ns#`) the template for subject, predicate and object (i.e., `ns:SubN`, `ns:PredN` and `ObjN`) and their type (i.e., URI, URI and Literal).

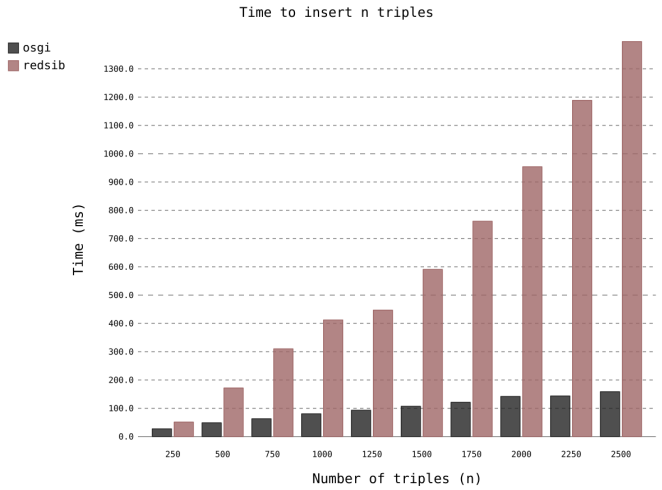

Fig. 3. The Update test performed on RedSIB and OSGi SIB

An example of the textual output represented by the related CSV file is shown below. The first field identifies the SIB, the second the number of triples. After the list of the collected values, the mean value, the minimum and maximum values and the variance are reported.

```
osgi,250,28.14,..,28.38,28.14,28.82,0.09
osgi,500,53.28,..,56.29,56.10,59.00,4.09
osgi,750,66.93,..,65.77,62.72,67.67,4.74
osgi,1000,78.53,..,77.30,76.37,78.53,0.82
osgi,1250,88.98,..,87.56,85.95,88.98,1.55
...
```

Fig. 4 shows the results of repeating the update test with only 1000 triples to insert on both the SIBs. The different kind of graph is the result of a different configuration set up by the user in the configuration manager.

B. Query Test

Two further tests have been performed on the OSGi SIB and on RedSIB, both consisting in retrieving the entire content of the knowledge base with RDF-M3 (Fig. 5) and with SPARQL (Fig. 6). With RDF-M3 the triple pattern is ($\star$, $\star$, $\star$), while the SPARQL query is:

```
SELECT ?s ?p ?o
WHERE {
```

Fig. 4. The Update test performed on RedSIB and OSGi SIB with box chart output

```
    ?s ?p ?o
}
```

The two charts allow to identify a better behavior of the RDF-M3 queries with respect to the given use case and, in general, a higher timeliness of the OSGi SIB.

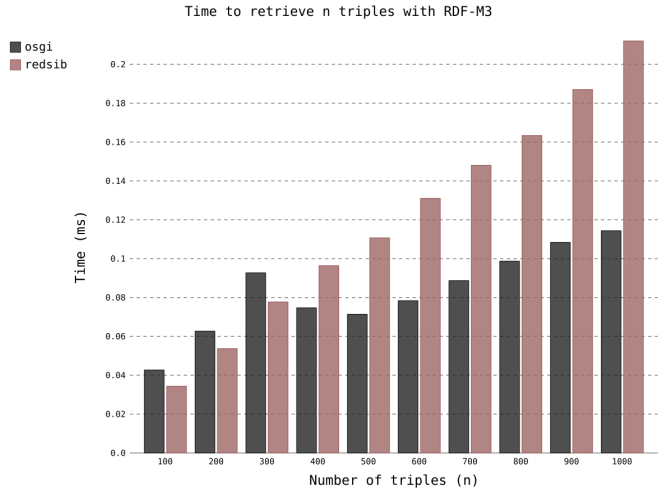

Fig. 5. The Query test performed on RedSIB and OSGi SIB to retrieve the entire RDF store content using the RDF-M3 query formalism

C. Subscription Test

The most relevant contribution of the PES, as mentioned in the introduction, is the ability to measure the time needed to receive a notification about an update of the RDF store. An example of the Subscription Test is shown in Fig. 7. In this example, a reference scenario of electric mobility was used and the time required to receive a notification about the registration of a new user was measured. In the chart is possible to observe the minimum notification time (in milliseconds), the maximum one and the mean value.

 ISBN: 978-1-61208-505-0

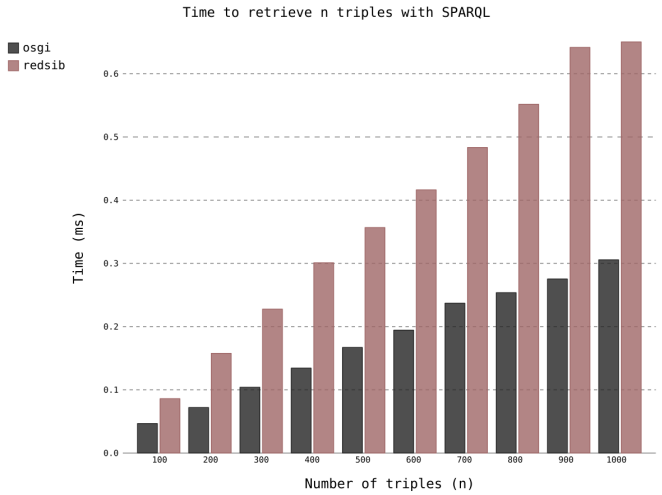

Fig. 6. The Query test performed on RedSIB and OSGi SIB to retrieve the entire RDF store content using the SPARQL query formalism

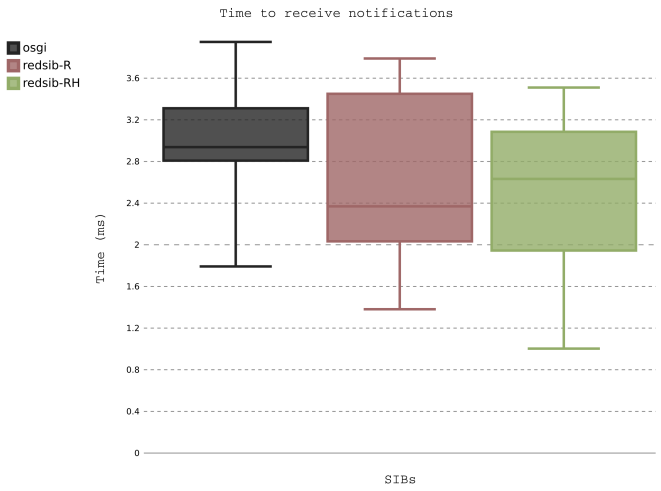

Fig. 7. The Subscription Test performed on RedSIB (with options `--ram` and `--ram-hash`) and OSGi SIB

It is possible to observe the presence in the box chart of two different boxes for RedSIB: the one labeled with `redsib-R` is related to the volatile storage without support for hash tables, while `redsib-RH` allows to test RedSIB with a volatile storage exploiting these data structures. The results show a little performance improvement in subscription management if the hash tables are used.
The SPARQL update is:

```
PREFIX ns: <http://.../ioe-ontology.owl#>
PREFIX rdf: <http://.../22-rdf-syntax-ns#>
INSERT DATA {
ns:User1_URI rdf:type ns:Person .
    ns:User1_URI ns:hasName "User Name" .
    ns:User1_URI ns:hasPasswd "UserPasswd"
}
```

while SPARQL subscription is:

```
PREFIX ns: <http://.../ioe-ontology.owl#>
PREFIX rdf: <http://.../22-rdf-syntax-ns#>
SELECT ?s
WHERE {
    ?s rdf:type ns:Person
}
```

VI. Conclusion and Future Work

A suite of software modules, named Performance Evaluation Suite (PES), aiming at evaluating the performance of Semantic Publish-Subscribe Message-Oriented Middlewares (SPS-MoMs) has been presented. A first implementation of the PES has been used to evaluate and compare the performance of two instances of the Smart-M3 Semantic Information Broker (SIB). The proposed platform allows to run existing benchmarks (e.g., SP^{2}B or LUBM) and to extend these benchmarks to evaluate the subscription mechanism provided by the broker. Future works include the definition of a set of benchmarks specific for IoT scenarios that can be used to perform an extensive performance analysis of available SPS-MoMs, including the several SIB implementations. The PES will be extended to provide a set of Key Performance Indicators (e.g., the average number of updates, or queries or subscriptions processed per unit time) that, along with current statistical analysis of the timing components, will allow to identify bottlenecks and limits of current SPS-MoMs driving the research towards an efficient and effective IoT solution.

References

[1] R. Minerva, A. Biru, and D. Rotondi, "Towards a definition of the internet of things (iot)," May 2015. [Online]. Available: http://iot.ieee.org/definition.html, [retrieved: April 2016]

[2] A. Asin and D. Gascon, "50 sensor applications for a smarter world," *Libelium Comunicaciones Distribuidas, Tech. Rep*, 2012.

[3] X.-Y. Liu and M.-Y. Wu, "Vehicular cps: an application of iot in vehicular networks," *Jisuanji Yingyong/ Journal of Computer Applications*, vol. 32, no. 4, pp. 900–904, 2012.

[4] W. He, G. Yan, and L. Da Xu, "Developing vehicular data cloud services in the iot environment," *Industrial Informatics, IEEE Transactions on*, vol. 10, no. 2, pp. 1587–1595, 2014.

[5] P. Ferreira, R. Martinho, and D. Domingos, "Iot-aware business processes for logistics: limitations of current approaches," in *Inforum*, vol. 3, 2010, pp. 612–613.

[6] S. Amendola, R. Lodato, S. Manzari, C. Occhiuzzi, and G. Marrocco, "Rfid technology for iot-based personal healthcare in smart spaces," *IEEE Internet of Things Journal*, vol. 1, no. 2, pp. 144–152, April 2014.

[7] C. Doukas and I. Maglogiannis, "Bringing iot and cloud computing towards pervasive healthcare," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*, July 2012, pp. 922–926.

[8] C. L. Hu, H. T. Huang, C. L. Lin, N. H. M. Anh, Y. Y. Su, and P. C. Liu, "Design and implementation of media content sharing services in home-based iot networks," in *Parallel and Distributed Systems (ICPADS), 2013 International Conference on*, Dec 2013, pp. 605–610.

[9] J. Burrell, T. Brooke, and R. Beckwith, "Vineyard computing: Sensor networks in agricultural production," *Pervasive Computing, IEEE*, vol. 3, no. 1, pp. 38–45, 2004.

[10] Z. Liqiang, Y. Shouyi, L. Leibo, Z. Zhen, and W. Shaojun, "A crop monitoring system based on wireless sensor network," *Procedia Environmental Sciences*, vol. 11, pp. 558–565, 2011.

 ISBN: 978-1-61208-505-0

[11] G. T. Costanzo, G. Zhu, M. F. Anjos, and G. Savard, "A system architecture for autonomous demand side load management in smart buildings," *IEEE Transactions on Smart Grid*, vol. 3, no. 4, pp. 2157–2165, Dec 2012.

[12] L. Schor, P. Sommer, and R. Wattenhofer, "Towards a zero-configuration wireless sensor network architecture for smart buildings," in *Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, ser. BuildSys '09. New York, NY, USA: ACM, 2009, pp. 31–36. [Online]. Available: http://doi.acm.org/10.1145/1810279.1810287

[13] M. Weyn, G. Ergeerts, L. Wante, C. Vercauteren, and P. Hellinckx, "Survey of the dash7 alliance protocol for 433mhz wireless sensor communication," *International Journal of Distributed Sensor Networks*, 2013.

[14] E. Kim, D. Kaspar, D. Gomez, and C. Bormann, "Problem statement and requirements for ipv6 over low-power wireless personal area network (6lowpan) routing," RFC 6606, October 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc6606.txt

[15] A. Banks and R. Gupta, "Mqtt version 3.1.1," October 2014, [retrieved: April 2016]. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html. Latest version: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html.

[16] Z. Shelby, K. Hartke, and C. Bormann, "The constrained application protocol (coap)," RFC 7252, June 2014. [Online]. Available: https://tools.ietf.org/html/rfc7252

[17] P. Saint-Andre, "Extensible Messaging and Presence Protocol (XMPP): Core," RFC 6120, October 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc6120.txt

[18] "Advanced message queuing protocol (amqp) version 1.0. 29," October 2012, [retrieved: April 2016]. [Online]. Available: http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-complete-v1.0-os.pdf

[19] A. D'Elia, L. Perilli, F. Viola, L. Roffia, F. Antoniazzi, R. Canegallo, and T. S. Cinotti, "A self-powered wsan for energy efficient heat distribution," in *2016 IEEE Sensors Applications Symposium (SAS)*, April 2016, pp. 1–6.

[20] M. Albano, L. L. Ferreira, L. M. Pinho, and A. R. Alkhawaja, "Message-oriented middleware for smart grids," *Computer Standards & Interfaces*, vol. 38, pp. 133–143, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0920548914000804

[21] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," pp. 34–43, 2001.

[22] "Rdf 1.1 concepts and abstract syntax," February 2014, [retrieved: April 2016]. [Online]. Available: https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/

[23] "Owl 2 web ontology language primer (second edition)," December 2012, [retrieved: April 2016]. [Online]. Available: https://www.w3.org/TR/2012/REC-owl2-primer-20121211/

[24] "Sparql 1.1 overview," March 2013, [retrieved: April 2016]. [Online]. Available: https://www.w3.org/TR/sparql11-overview/

[25] Y. Guo, Z. Pan, and J. Heflin, "LUBM: A benchmark for OWL knowledge base systems," *Web Semantics*, vol. 3, no. 2-3, pp. 158–182, 2005.

[26] Y. Guo, A. Qasem, Z. Pan, and J. Heflin, "A requirements driven framework for benchmarking Semantic Web knowledge base systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 2, pp. 297–309, 2007.

[27] R. Garcìa-Castro and E. al., *Web Semantics: Science, Services and Agents on the World Wide Web, Special Issue on Evaluation of Semantic Technologies*. Elsevier, 2013, vol. 21.

[28] C. Bizer and A. Schultz, "The Berlin SPARQL Benchmark," *International Journal on Semantic Web & Information Systems*, vol. 5, no. 2, pp. 1–24, 2009.

[29] M. Murth, D. Winkler, S. Biffl, E. Kühn, and T. Moser, "Performance Testing of Semantic Publish / Subscribe Systems," *Journal Of Web Semantics*, pp. 45–46, 2010.

[30] M. Murth, "K{ü}hn, e.: A Semantic Event Notification Service for Knowledge-Driven Coordination," in *Proc. of 1st Int'l. workshop on emergent semantics and cooperation in open systems (ESTEEM), co-operation with the 2nd Int'l. Conf. on Distributed Event-Based Systems (DEBS 2008), Rome, Italy*, 2008.

[31] J. Honkola, H. Laine, R. Brown, and O. Tyrkko, "Smart-M3 information sharing platform," in *The IEEE symposium on Computers and Communications*. IEEE, 2010, pp. 1041–1046.

[32] D. Manzaroli, L. Roffia, T. S. Cinotti, E. Ovaska, P. Azzoni, V. Nannini, and S. Mattarozzi, "Smart-m3 and osgi: The interoperability platform," in *Computers and Communications (ISCC), 2010 IEEE Symposium on*. IEEE, 2010, pp. 1053–1058.

[33] "Redsib," [retrieved: April 2016]. [Online]. Available: https://sourceforge.net/projects/smart-m3/files/Smart-M3-RedSIB_0.9.2/

[34] Y. Guo, A. Qasem, Z. Pan, and J. Heflin, "A requirements driven framework for benchmarking semantic web knowledge base systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 2, pp. 297–309, 2007.

[35] Y. Guo, Z. Pan, and J. Heflin, "Lubm: A benchmark for owl knowledge base systems," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 3, no. 2, pp. 158–182, 2005.

[36] M. Schmidt, T. Hornung, G. Lausen, and C. Pinkel, "Sp$\hat{2}$bench: A sparql performance benchmark," in *2009 IEEE 25th International Conference on Data Engineering*, March 2009, pp. 222–233.

[37] M. Ley, "The dblp computer science bibliography: Evolution, research issues, perspectives," in *String Processing and Information Retrieval*. Springer, 2002, pp. 1–10.

[38] T. Berners-Lee and D. Connolly, "Notation3 (n3): A readable rdf syntax," *W3C Team Submission: http://www. w3. org/TeamSubmission*, no. 3, 1998.

[39] C. Bizer and A. Schultz, "The berlin sparql benchmark," 2009.

[40] M. Murth, D. Winkler, S. Biffl, E. Kühn, and T. Moser, "Performance testing of semantic publish/subscribe systems," in *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer, 2010, pp. 45–46.

[41] "Ioe." [Online]. Available: http://www.artemis-ioe.eu/ [retrieved: April 2016]

[42] "Arrowhead ahead of the future," http://www.arrowhead.eu/ [retrieved: April 2016].

[43] "Chiron." [Online]. Available: http://www.unibo.it/en/research/projects-and-initiatives/unibo-projects-under-7th-framework-programme/cooperation-1/information-and-communication-technology-ict-1/chiron [retrieved: April 2016]

[44] A. D'Elia, F. Viola, F. Montori, M. Di Felice, L. Bedogni, L. Bononi, A. Borghetti, P. Azzoni, P. Bellavista, D. Tarchi *et al.*, "Impact of interdisciplinary research on planning, running, and managing electromobility as a smart grid extension," *Access, IEEE*, vol. 3, pp. 2281–2305, 2015.

[45] A. Varfolomeyev, D. Korzun, A. Ivanovs, and O. Petrina, "Smart personal assistant for historical tourism," in *RECENT ADVANCES in ENVIRONMENTAL SCIENCES and FINANCIAL DEVELOPMENT. 9-15 Nov, 2014*, 2014, p. 9.

[46] A. Smirnov, A. Kashevnik, S. I. Balandin, and S. Laizane, "Intelligent mobile tourist guide," in *Internet of Things, Smart Spaces, and Next Generation Networking*. Springer Berlin Heidelberg, 2013, pp. 94–106.

[47] F. Morandi, L. Roffia, A. DElia, F. Vergari, and T. S. Cinotti, "Redsib: a smart-m3 semantic information broker implementation," in *Proc. 12th Conf. of Open Innovations Association FRUCT and Seminar on e-Tourism*. SUAI, 2012, pp. 86–98.

[48] F. Viola, A. D'Elia, L. Roffia, and T. Salmon Cinotti, "A modular lightweight implementation of the smart-m3 semantic information broker," in *18th FRUCT*, 2016, pp. 370–377.

[49] I. V. Galov, A. A. Lomov, and D. G. Korzun, "Design of semantic information broker for localized computing environments in the internet of things," in *Open Innovations Association (FRUCT), 2015 17TH Conference of*. IEEE, 2015, pp. 36–43.

[50] D. Crocker, "Standard for the format of ARPA Internet text messages," Internet Requests for Comments, RFC Editor, RFC 822, August 1982. [Online]. Available: https://tools.ietf.org/html/rfc822

 ISBN: 978-1-61208-505-0