Research online Publishing www.ronpub.com

Open Access

Open Journal of Internet of Things (OJIOT)
Volume 8, Issue 1, 2022
http://www.ronpub.com/ojiot
ISSN 2364-7108

WoTHive: Enabling Syntactic and Semantic Discovery in the Web of Things

Andrea Cimmino, Raúl García-Castro

Universidad Politécnica de Madrid, C. Ramiro de Maeztu, 7, 28040, Madrid, Spain
{andreajesus.cimmino, r.garcia}@upm.es

Abstract

In the last decade the Internet of Things (IoT) has experienced a significant growth and its adoption has become ubiquitous in either business and private life. As a result, several initiatives have emerged for addressing specific challenges and provide a standard or a specification to address them; like CoRE, Web of Things (WoT), oneM2M, or OGC among others. One of these challenges revolves around the discovery procedures to find IoT devices within IoT infrastructures and whether the discovery performed is semantic or syntactic. This article focusses on the WoT initiative and reports the benefits that Semantic Web technologies bring to discovery in WoT. In particular, one of the implementations for the WoT discovery is presented, which is named WoTHive and provides syntactic and semantic discovery capabilities. WoTHive is the only candidate implementation that addresses at the same time the syntactic and semantic functionalities specified in the discovery described by WoT. Several experiments have been carried out to test WoTHive; these advocate that the implementation is technically sound for CRUD operations and that its semantic discovery outperforms the syntactic one implemented. Furthermore, an experiment has been carried out to compare whether syntactic discovery is faster than semantic discovery using the Link Smart implementation for syntactic discovery and WoTHive for semantic.

Type of Paper and Keywords

Short communication: *Internet of Things, Web of Things, directory-based discovery*

1 Introduction

In the last decade, the Internet of Things (IoT) has become ubiquitous in both business and everyday life. The context in which IoT devices are used is wide and broad, fostering the consolidation of initiatives that aim to establish the interactions of these IoT devices. Some standard initiatives are Constrained RESTful Environments (CoRE) [25], XMPP [30], HyperCat specification, OGC Sensor Observation Service [5], Web of Things [9], or oneM2M [26]; other well-known initiatives are Vorto [7].

One of the main goals of these initiatives is to define discovery procedures for finding existing IoT devices within IoT infrastructures. Although there are several generic discovery approaches [4], one of the most adopted by these initiatives is directory-base discovery. It consists of having a service named *resource directory* in which clients can register documents called *resource descriptions* that provide information about an IoT device or an IoT infrastructure. The nature of the information contained in *resource descriptions* depends on the initiative; it can be very different: endpoints where data can be fetched, meta-data (such

This paper is accepted at the *International Workshop on Very Large Internet of Things (VLIoT 2022)* in conjunction with the VLDB 2022 conference in Sydney, Australia. The proceedings of VLIoT@VLDB 2022 are published in the Open Journal of Internet of Things (OJIOT) as special issue.

as location, vendor, etc.), information about security protocols used, or validation information (such as JSON schema). Then, the discovery is performed by issuing a discovery criterion to the *resource directory*, which finds the *resource descriptions* that meet the criterion and answers with this result.

The different initiatives specify the format and model that *resource descriptions* must have, the format of the discovery criterion and its results, and the protocol and API that a *resource directory* must provide. When the models and format of the *resource descriptions* rely on W3C standards from the Semantic Web, such as RDF and ontologies, and the discovery criterion is based on SPARQL; the discovery is named semantic discovery. In contrast, when the format and model rely on others that are non-semantic, the discovery is known as syntactic discovery. The Semantic Web technologies improve different aspects of the discovery approaches defined by the different initiatives; a good example of these benefits has been reported for the discovery procedure of oneM2M [17, 12].

This article focuses on the Web of Things (WoT) initiative and describes how Semantic Web technologies foster and improve the discovery procedure defined by this initiative. In particular, the WoTHive *resource directory* that implements the current WoT discovery procedure is presented. This directory provides syntactic and semantic capabilities; which are used in the article for showing the benefits and drawbacks of relying on semantic or syntactic discovery. The WoTHive has been tested with several experiments, among which, one of them aims at comparing the potential overhead that semantic discovery may entail in comparison with the syntactic discovery for equivalent discovery criterion.

The rest of this article is structured as follows. Section 2 presents an analysis of proposals from the literature; Section 3 introduces the WoTHive implementation of the WoT discovery; Section 5 presents some results of the experiments carried out in the article; finally, Section 6 presents conclusions.

2 Related Work

The different initiatives have spent considerable effort defining or standardising the different elements that revolve around IoT, and specifically, for discovery: i) the resource directories APIs; ii) resource description format and model; iii) the discovery procedure; and iv) the discovery criterion and its results. Of all these initiatives, some are well-known standards [4]: Constrained RESTful Environments (CoRE) [23], XMPP (Internet of Things Discovery) [30], HyperCat specification [21], OGC Sensor Observation Service [5], WoT [9], and oneM2M [26]. Other initiatives that are widely used are not standards, such as Eclipse VORTO.

All initiatives can be grouped depending on their focus: some only establish the data format and model of the resource descriptions to describe IoT devices or infrastructures (SAREF, SSN, VORTO, HyperCat); instead, others describe the API of their resource directory and also how the discovery procedure must function: how to receive the discovery criterion and generate its result. Table 1 summarises the contributions of the different initiatives.

2.1 Discovery in IoT Initiatives

The IETF CoRe Resource Directory [25] is a specification based on the CoAP protocol. The specification expects that IoT infrastructures submit a *resource description* expressed in the CoRE Link Format [23], containing a set of attributes[1] such as the type of the endpoint or the *time-to-live* of the description submitted. In addition, the CoRe Resource Directory has a discovery interface to which search criteria referencing some attributes are submitted. As a result, the discovery interface outputs a set of resource descriptions that meet the search criterion submitted.

XMPP Internet of Things Discovery [30] is a specification for peer-to-peer networks that connect IoT infrastructures. In this specification, *resource directory* is also known as Thing Registry, which expects a template-based XML document as a resource description. The specification allows those restrictions to contain a set of tags to specify metadata such as location[2]. The discovery criterion supported by a Thing Registry is a URL request with a set of search operators that can be combined with AND or OR operators. [3]

HyperCat [16] is based on the concept of the IoT catalogue, that is, a *resource directory*. This component assumes that each IoT infrastructure is internally identified by a unique URI associated with a JSON document, that is, the *resource description*. The *resource descriptions* rely on the HyperCat model that is designed to be translated into RDF when required [28, 27]. The discovery criteria of this specification are URI requests with a set of parameters which are used to filter the descriptions or to find suitable URIs whose descriptions fulfil a criterion. It uses Media Types for Sensor Markup Language (SENML) to describe metadata [11].

[1] https://tools.ietf.org/html/draft-ietf-core-resource-directory-21#page-22

[2] https://xmpp.org/extensions/xep-0347.html#tags

[3] https://xmpp.org/extensions/xep-0347.html#search

Table 1: Initiatives from the literature specifying IoT discovery elements

Initiative	Standard	Resource Directory API	Resource Description		Discovery Specification		
			Format	Model	Protocol	Criterion	Result
CoRE [23]	Yes	CoRE Resource Directory [25]	CoRE Link (Web Link [20])	List of link parameters	Any binding supporting URIs	CoRE Link (URI with attributes)	Set of CoRE links (URIs)
XMPP [30]	Yes	Thing Registry	XML	List of XML tags	XMPP	XML document with comparison operators using the XML tags	Set of XML documents
HyperCat [16]	-	HyperCat Catalogue	Json	Catalogue Object	HTTP	URI with attributes	Single Json document
SenML [11]	Yes	-	Json, XML, or EXI	List of Media Types	-	-	-
OGC	Yes	Sensor Observation Service [5]	XML	SensorML [2]	Any binding supporting URIs	URI with attributes	Set of XML documents
WoT	Yes	Thing Description Directory (TDD)	Json-LD 1.1 (RDF)	Thing Description (TD)	HTTP	JsonPath, XPath or SPARQL	Set of Json-LD (TDs)
OneM2M	Yes	Common Services Entity (CSE)	RDF	oneM2M Ontology	CoAP, MQTT, HTTP and Web Socket	CoRE Link (URI with attributes) or SPARQL	Set of URIs
SAREF	Yes	-	RDF	SAREF Ontology	-	-	-
SSN	Yes	-	RDF	SSN Ontology	-	-	-
CoAP [24]	Yes	CoRE Resource Directory [25]	CoRE Link (Web Link [20])	List of link parameters	CoAP	CoRE Link (URI with attributes)	Set of CoRE links (URIs)
VORTO [7]	-	-	Vorto Language	-	-	-	-
OGC	Yes	Catalogue Service [19]	XML	Catalogue Schema	HTTP	CommonQL	Set of XML documents
OAS	-	-	Json	OpenAPI Language	-	-	-

OGC Sensor Observation Service (SOS) [5] is a specification designed for IoT infrastructures on the Web. In this specification, resource descriptions must be expressed according to the Sensor Model Language. (SensorML) [2]; these are RDF expressed as XML documents that contain infrastructure metadata and also allow for the injection of their data values. The OGC discovery criterion in this specification is URI requests with parameters; however, SPARQL queries could also be used [10].

The Web of Things (WoT) [13] is a Web-based specification. It relies on Thing Description Directories, that is, resource directories that expect IoT infrastructures to submit a Thing Description [13], that is, a resource description, expressed in JSON-LD.1.1 [14] containing related meta-data. Discovery criteria are SPARQL queries, filtering criteria such as Json Path [3] or XPath [1], and URI requests with attributes. Nevertheless, the WoT discovery standard is currently under development, and thus, some minor decisions may change over time; which is not the case for the resource descriptions (which are a standard) and the API of the resource directory.

The oneM2M specification [26] is based on the concept of CSE as *resource directory*. CSEs are endowed to provide two levels of discovery. On the one hand, IoT infrastructures can register in a CSE by submitting a URI with parameters similar to the CoRE Link Format [23]. On the other hand, IoT infrastructures can register in a CSE by submitting an RDF document expressed according to the oneM2M ontology (TS-0012). Then, discovery is carried out differently (TR-0045), for the former case with a URI request with parameters and for the latter with a SPARQL query.

The OpenAPI Specification (OAS) defines a standard programming language-agnostic interface to RESTful APIs that allows both humans and computers to discover and understand the capabilities of the service without access to source code, documentation, or through network traffic inspection. When properly defined, a consumer can understand and interact with the remote service with a minimal amount of implementation logic. An OpenAPI definition can then be used by documentation generation tools to display the API, code generation tools to generate servers and clients in various programming languages, testing tools, and many other use cases.

2.2 Discovery in WoT

During the last decade, the World Wide Web Consortium has been working on the Web of Things (WoT) specification. Recently, this specification has published two standards: Thing Description (TD) [13], which describes metadata and interfaces of physical or virtual entities that provide interaction endpoints, and WoT abstract architecture [15], which is a conceptual framework that defines a set of modular building blocks and how they interact [15] .

WoT has developed a large number of proposals that aim to provide capabilities for different research topics. In all likelihood, one of the most addressed topics is the Thing Description discovery that aims at filtering a collection of Thing Description for a given discovery criterion [31]. Nevertheless, the lack of a standard approach for discovery from the WoT working group has led to many different proposals [18].

Recently, the divergence of discovery proposals motivated the WoT to promote a new group focused on standardising the discovery in WoT. The building block that implements discovery is the Thing Description Directory (TDD), which follows a Resource Directory approach [6]. The TDD aims to store a collection of TDs and then provide users with discovery capabilities by means of three different discovery criterion [6]. However, there is no experimental analysis of the new implementations of WoT discovery in terms of efficiency and expressiveness.

The TDs promoted by WoT are expressed in JSON-LD 1.1 that allows to use procedures over the TDs, consider them as plain JSONs or as RDF. This entails that different implementations of the TDD may range from non-semantic based to full semantic based discovery. Due to this reason, the discovery promoted by the WoT can be either syntactic using JSON Path or XPATH expressions or semantic using SPARQL.

In the Web of Things, there are currently three proposals that aim to implement the current draft specification [4]. LinkSmart Thing Directory is provided by the University of Fraunhofer [29], it implements syntactic discovery based on JSON Path. Logilab TDD is provided by Siemens; it implements semantic discovery based on SPARQL. Finally, the WoTHive is provided by the Universidad Politécnica de Madrid and, to the authors' knowledge, is the only implementation covering the syntactic and semantic discovery relying on JSON Path and SPARQL. Furthermore, the WoTHive provides other semantic capabilities, such as syntactic and semantic validation, or discovery by federation.

3 THE WOTHIVE DIRECTORY

The WoTHive directory architecture is shown in Figure 1. As can be seen, the directory counts with four main components: the registration, discovery,

[4] https://github.com/w3c/wot-discovery/blob/main/implementations/README.md

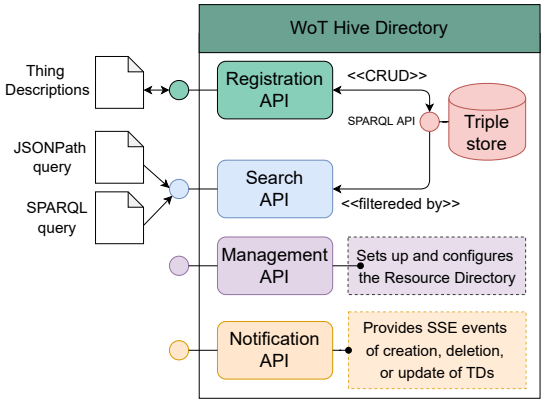


Figure 1: WoTHive architecture

notification, and management components. The former component provides the functions for creating, reading, updating, or deleting Thing Descriptions in JSON-LD 1.1 or Turtle formats. The discovery component implements the JSONPAth and SPARQL based discovery. The notification component provides an event-based endpoint following the Server-Sends Events (SSE) standard that informs about the operations performed by the registration component. Finally, the Management component allows a user to set up several functionalities of the directory (like the connection to the triple store). Note that WoTHive does not provide any security mechanisms; the reason behind this decision is the fact that this service is usually used in a context where security is handled by a third-party service (like NGINX). The current WoTHive implementation is publicly available on GitHub[5].

The following subsections report detailed information on the registration and discovery APIs and how they function to provide semantic and syntactic capabilities. The *Management API* and *Notification API* are not explained because they follow a standard (such as *Notification API*) or fall outside of the scope of this article (such as *Management* API).

3.1 Registration API

The registration API enables four main operations: 1) creation of TDs, which receives a TD in JSON-LD 1.1 format, enriches it with some registration information (e.g., creation date) and stores it into the triple store as RDF triples; 2) Anonymous creation of TDs, which only difference from the first functionality is the fact that for the first operation an *id* must be provided, whereas for anonymous creation is the WoTHive, which automatically assigns an *id* to the TD; 3) Update existing TD, which can be achieved either replacing the TD with a specific id (PUT) or by performing a patch operation (PATCH): 4) Deleting an existing TD, which removes the TD from the system; and finally, 5) Retrieve TDs, which can be performed individually by providing a TD id or listing all the TDs stored. Notice that performing all these operations requires translating the TDs from JSON-LD 1.1 to triples, and back, ensuring that all the triples from one TD are always correctly handled by the triple store.

In general, the operations that may receive either a JSON-LD 1.1 or an RDF document expressed in another serialisation follow the same steps; for instance, registering a TD: 1) if the TD is expressed in JSON-LD 1.1 the validation using JSON schema is applied; then the document is translated into the RDF serialisation N3; 2) if the TD is expressed in any other RDF serialisation than JSON-LD 1.1, then the RDF is serialised into N3; 3) then, the WoTHive validates the TD expressed in N3 using SHACL; 3) after which the TD in N3 is translated into an INSERT SPARQL query; 4) Finally, the query is sent to the triple store. Note that validations are performed as long as the user chooses them to occur (this feature can be enabled or disabled using the *Management API*).

The strategy that WoTHive follows for storing the TDs and ensuring the correctness of the operations is based on named graphs. The triple store allows users to define a label to mark the different triples stored; therefore, all triples obtained as a result of receiving a TD are marked with its associated *id* as the label, that is, the name of the graph. In addition, the WoTHive creates a special named graph that stores information about the TDs stored, for instance, if they had specifically written the td:Thing (since this information is paramount for translating back from triplets to JSON-LD 1.1).

As a result, all the previous operations are actually performed, namely over the named graphs. For example, a deletion operation consists in uniquely deleting the named graph marked with the TD *id*. Another example is to read a TD, which consists of reading the triplets specified in the named graph with the *id* of such a TD and translating them to JSON-LD 1.1 if needed. In the operation of registering a TD described previously, the SPARQL query indicates the named graph.

3.2 Discovery

The WoTHive implements both syntactic and semantic discovery. The WoT specification for directories allows syntactic discovery to be implemented using either JSON Path or XPath, or both. WoTHive implements only the JSON Path, which better suits the TDs since they are usually expressed in JSON-LD 1.1. To solve the JSON

[5] https://github.com/oeg-upm/wot-hive

Path discovery criterion, WoTHive has to retrieve all the TDs stored in the triple store into memory, translate them into JSON-LD 1.1, and then solve the filtering expression.

The WoT specification for directories mandates that semantic discovery must be implemented with SPARQL, both the protocol and the query language. To this end, the WoTHive counts with a triple store for keeping the TDs. In addition, when a SPARQL query is issued, the WoTHive delegates solving the query directly to the triple store. In particular, the WoTHive implementation is distributed with a Jena Fuseky[6].

As a result, the syntactic discovery is not as suitable as the semantic one, since the former is performed in memory (thus, it is fast but memory consuming); instead, the semantic relies on a triple store, which are designed for solving queries in an optimum time.

4 Benefits of Semantic Web Technologies for Discovery

Once the WoTHive implementation is described, the authors would like to mark some common discussion points worth mentioning between syntactic and semantic discovery and how the Semantic Web brings benefits to the discovery.

Discovery based on standard language and procedures: IoT moves to the adoption of standards to countermeasure existing heterogeneity in different aspects, such as interoperability, and therefore the discovery of IoT should also move in that direction. The Semantic Web offers standards like SPARQL that is either a W3C standard query language [22] and also a W3C standard protocol [8]. Note that XPath is only a standard query language [1] and JSON Path is not a standard.

Query language vs filtering language: JSON Path and XPath are not query languages themselves, but rather filtering languages for tree-based documents; JSON and XML, respectively. As a result, JSON Path and XPath are suitable for filtering documents based on some attributes at the same tree-level; however, they are limited for retrieving specific elements of the same document when there is more than one filter at different tree levels. In addition, complex queries with special functions or aggregating metrics are not possible to express in those filtering languages. Instead, SPARQL is a query language that lacks these limitations.

Query federation: the IoT nature tends to be distributed, and to this end, the discovery should also adapt to this nature. JSON Path and XPath are not suitable for performing distributed discovery among different WoT directories, and thus do not encompases the distributed nature of IoT. On the contrary, the Semantic Web also has a distributed nature, which is reflected in SPARQL with the federation mechanism. SPARQL queries may specify multiple endpoints where such a query must be solved and automatically return an aggregation of the results obtained; this operation is specified in the SPARQL standard and is known as SPARQL federation. Since WoTHive implements a standard SPARQL endpoint, it also provides the capability of solving the SPARQL query among a set of other SPARQL endpoints that can be either WoTHive directories or any other SPARQL endpoint.

[6] https://jena.apache.org/documentation/fuseki2/

5 Experiments

This section aims to provide an empirical set of experiments to analyse the performance of WoTHive APIs for registration and discovery (section 5.1 and subsection 5.2), and also to study the difference between the syntactic and semantic discovery defined by WoT (section 5.3). All experiments have been carried out using the WoTHive Docker recipe[7] version 2.5.0. Furthermore, all scripts and materials used are available at Zenodo [8]. These experiments have been carried out on an Os X with 2,5 GHz of Intel Core i7 and 16GB of RAM.

5.1 WoTHive Registration

This experiment aims to test the *Registration API*. To this end, two Thing Descriptions have been defined; one is anonymous and the other has a known *id*. Using both and the Taurus software [9], most of the registration operations have been tested (create, create anonymous, delete, and retrieve). Operations are performed sending requests during 1 minute, after which, there is a ramp-up to 50 during 4 minutes; in other words, during 4 minutes the number of concurrent requests for each operation grows up to 50. Figure 2 shows the results achieved by WoTHive in this experiment.

The y-axis shows the number of virtual users using the application (vu) and the response times in milliseconds (ms). The x-axis shows the 5 minutes of execution. The requests for creating a TD is called in the legend REGISTER, and the anonymous registration REGISTER_ANON; instead, the retrieving operation is called GET_TD and the deletion is DELETE_TD. Finally, ALL corresponds to the number of concurrent users performing one of these operations. As a result, it can be concluded from the results shown in Figure 2 that

[7] https://github.com/oeg-upm/wot-hive
[8] https://doi.org/10.5281/zenodo.6674151
[9] https://gettaurus.org/

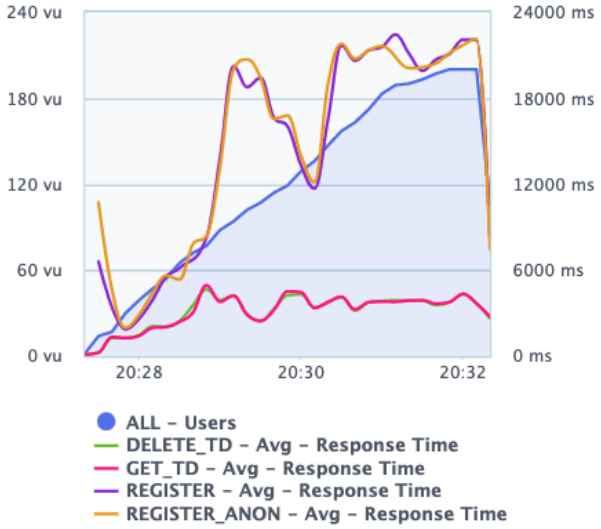


Figure 2: Registration, retrieval, and deletion of TDs in WoTHive

WoTHive supports a large number of parallel operations from the Registration API.

5.2 WoTHive Discovery

The WoTHive implementation has especially been endowed with semantic capabilities, whereas the implementation is not optimised for syntactic discovery. For this reason, the syntactic discovery is likely to perform worse than the semantic discovery. The goal of this experiment is to find the limitations of syntactic discovery and compare them with semantic discovery. To this end, 10 queries are defined, 5 are JSON Path queries, and the other 5 are equivalent SPARQL queries, reported in Table 3 and Table 4, respectively. Notice that the SPARQL queries cover a wide number of types: SELECT, DESCRIBE and CONSTRUCT.

The experiment consists in first populating the WoTHive directory with 25 TDs, then all the queries are solved 8 times each, and the first three results are discarded as warm-up, with the remaining 5 the average response time being computed. This process is then repeated 4 times until the number of TDs reaches 100. Figure 3 shows the results obtained for this experiment.

As mentioned above, and also shown in Figure 3, the syntactic discovery of WoTHive is designed to be used with small amounts of TDs. Instead, the semantic discovery is capable of dealing with a larger number of stored TDs. Note that an architecture optimised for syntactic discovery may have other results. To address this issue, Section 5.3 presents a comparison with an implementation that has an optimised syntactic discovery.

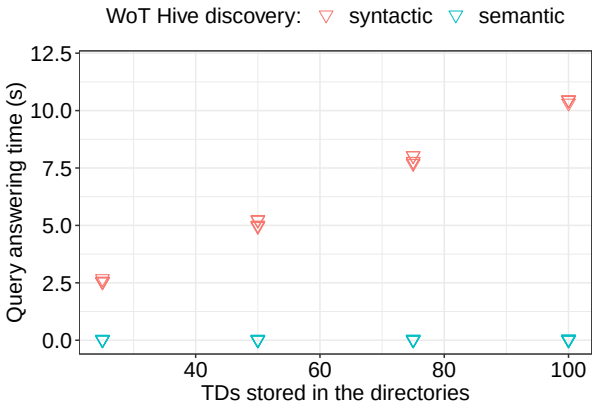


Figure 3: Response times of WoTHive for queries from Table 3 and Table 4

5.3 Syntactic vs Semantic Discovery

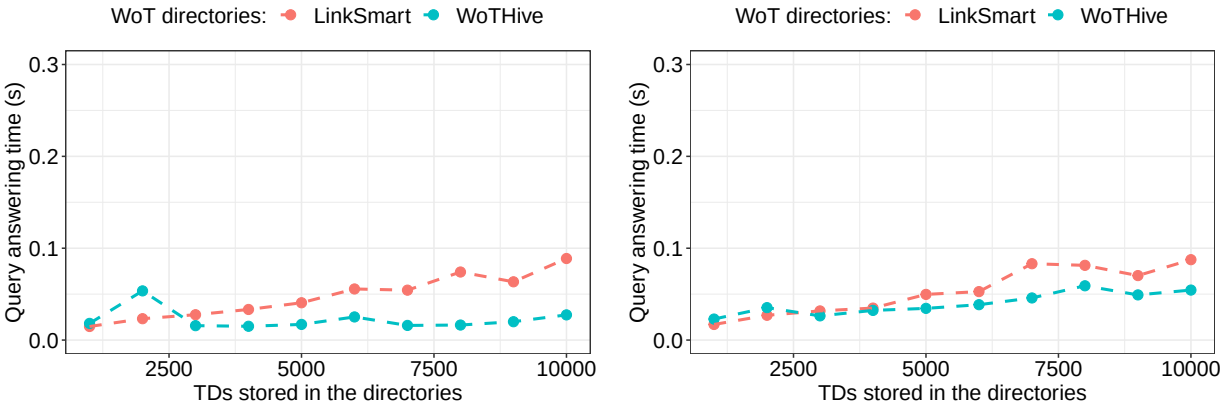
The goal of this experiment is to analyse the performance of the WoT syntactic and semantic discovery and to try to analyse which performs better and under which circumstances. Since the syntactic discovery of WoTHive is not suitable for large scenarios and for the sake of fairness, this experiment has been carried out using another of the candidate recommendations for the WoT Discovery; the LinkSmart directory[10], which is optimised for syntactic discovery. For this experiment, the queries reported in Table 3 and Table 4, respectively, are used. However, JSON Path queries are always sent to the LinkSmart directory; instead, SPARQL queries are sent to the WoTHive directory.

The experiment consists in first populating both directories with the same 1,000 TDs, then all the queries are solved 8 times each, and the first three results are discarded as warm-up, with the remaining 5 the average response time being computed. Then, this process is repeated 10 times until the number of TDs reaches 10,000. Figure 4 shows a subfigure for each query where the x axis reports the response time needed by LinkSmart and WoTHive and the y axis reports the number of TDs stored in each directory when a query criterion was solved.

Note that WoTHive obtains faster results for queries Q1 (sub-figure 4a), Q2 (sub-figure 4b), and Q4 (sub-figure 4d). These results are especially interesting, since they not only report faster response times, but also, as the number of TDs increases, that the trend line of WoTHive has a lower slope; this is especially relevant in Q4 (subfigure 4d). However, in Q3 (subfigure 4a) and Q5 (subfigure 4a) WoTHive gets worse results.

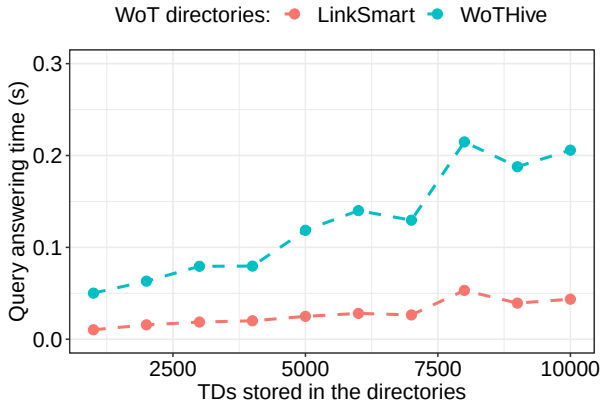
The reason behind these bad results of time is the

[10] https://github.com/linksmart/thing-directory

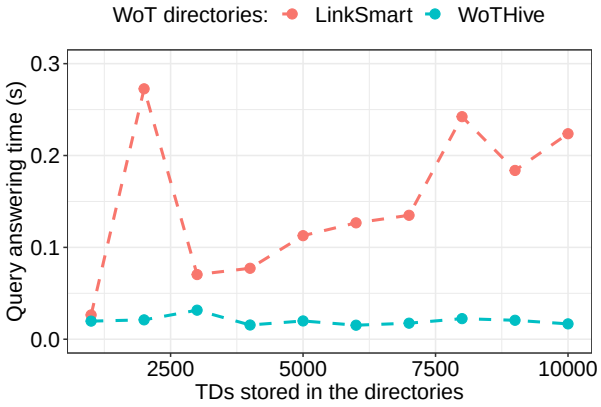


(a) Results for Q1 from Table 3 and Table 4

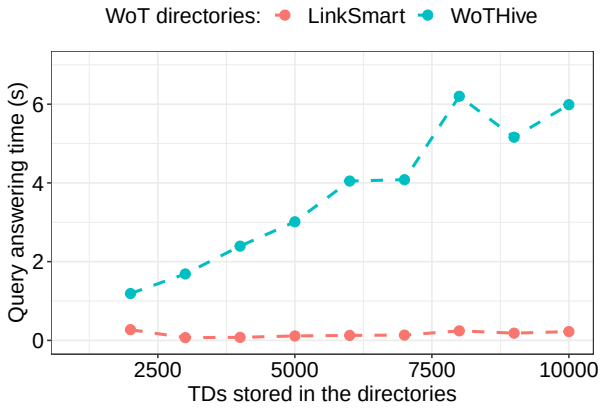
(b) Results for Q2 from Table 3 and Table 4



(c) Results for Q3 from Table 3 and Table 4



(d) Results for Q4 from Table 3 and Table 4



(e) Results for Q5 from Table 3 and Table 4

Figure 4: Discovery response times in seconds achieved by LinkSmart and WoTHive

Table 2: Query answer loads with 10,000 TDs stored

Query	LinkSmart answer size	WoTHive answer size
Q1	242 Bytes	740 Bytes
Q2	238 Bytes	740 Bytes
Q3	188 Bytes	1.04 Mega bytes
Q4	274 Bytes	2.12 Kilo bytes
Q5	6.38 Mega bytes	34.04 Mega bytes

size of the query answers; for this sake, the response sizes when 10,000 TDs are stored have been captured and reported in Table 2. For example, the result of Q5 (subfigure 4a) outputs all stored TDs; in the worst-case scenario LinkSmart returns 10,000 TDs expressed in JSON-LD 1.1 that are 6.38Mb in memory, instead, WoTHive returns 10,000 TDs expressed in N3 that are 34Mb in memory. The differences in the serialisations is the reason behind the bad results of WoTHive: JSON-LD 1.1 needs fewer characters than the N3 serialisation. This drawback will be resolved in the future when triple stores natively support JSON-LD 1.1.

From this experiment, it can be concluded that semantic discovery is as reliable as syntactic discovery. Although it is true that when the discovery response is large, the syntactic discovery tends to answer faster, the semantic discovery seems to scale up as well as the syntactic. In addition, for queries where the response size is simmilar, the results advocate that semantic discovery can be faster than syntactic discovery.

6 Conclusions

In this article, the WoTHive implementation for the Web of Things discovery directory is presented. This implementation aims to provide the syntactic and semantic features for discovery. Taking advantage of this dual nature, the article presents the benefits that semantic discovery provides and that syntactic lacks. The experiments carried out aim, on the one hand, at showing that the WoTHive is a solid implementation for performing the different operations, such as registration of TDs and discovery. However, the experiments carried out have analysed whether the semantic or syntactic discovery performs better and under what circumstances.

The experiments advocate that semantic discovery can be as fast and resilient as syntactic; in fact, for similar response sizes, semantic discovery performs better. Notice that in the experiments, the syntactic queries and their equivalent (yet not the same) queries are solved in different implementations of the WoT discovery directory. These results argue for the fact that semantic discovery is as efficient as syntactic discovery. RDF counts with databases that support native query-solving mechanisms that are highly optimised; instead, the JSON databases usually do not count with native and optimised JSON Path query-solving functionalities, meaning that when the data scale up, the response times will only increase. In fact, LinkSmart does the filtering on memory, whereas the WoTHive SPARQL solving mechanism actually delegates the whole operation to the triple store. However, note that the response times for both implementations are quite good and that the LinkSmart answers have an O(n) shape. Although the results of this experiment show that semantic discovery is faster than syntactic carried out in memory for some scenarios, if in future an implementation relies on a DB supporting JSON Path then in all likelihood the response times will be similar.

In addition, results have shown that semantic discovery performs worst when response sizes are larger due to the fact that SPARQL queries do not support JSON-LD 1.1 responses, the other RDF serialisations need a larger number of characters to express the same information than one using JSON-LD 1.1. For this reason, larger responses will take longer to compute.

In the future, the experimentation will be extended to add queries that rely on the federation and show how the overall WoT discovery can benefit from this mechanism that only exists for semantic discovery. In addition, to countermeasure the poor performance in the JSON Path achieved by WoTHive, a translation algorithm to translate from the JSON Path to SPARQL queries will be explored. Finally, two additional experiments will be introduced, one in which the results of the queries are compressed and the other in which the RDF is translated back to JSON-LD 1.1. The aforementioned experiments will show the overhead that RDF verbosity introduces and also the overhead of changing the serialisation to JSON-LD 1.1.

7 Acknowledgements

This work is partially funded by the European Union's Horizon 2020 Research and Innovation Programme through the AURORAL project, Grant Agreement No. 101016854. The authors would like to thanks Farshid Tavakoliadeh, the developer and author of LinkSmart, for his help and guidance in the comparison of the WoT directories.

References

[1] A. Berglund, S. Boag, D. Chamberlin, M. F. Fernández, M. Kay, J. Robie, and J. Siméon, "XML

Path Language (XPath) 2.0 (Second Edition)," *W3C Recommendation*, 2010.

[2] M. Botts and A. Robin, "OpenGIS® Sensor Model Language (SensorML) Implementation Specification. Version 1.0.0." Open Geospatial Consortium (OGC 07-000), 2007, p. 180.

[3] P. Bourhis, J. L. Reutter, F. Suárez, and D. Vrgoč, "JSON: data model, query languages and schema specification," in *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI symposium on principles of database systems*, 2017, pp. 123–135.

[4] A. Bröring, S. K. Datta, and C. Bonnet, "A Categorization of Discovery Technologies for the Internet of Things," in *Proceedings of the 6th International Conference on the Internet of Things*. ACM, 2016, pp. 131–139. [Online]. Available: http://dl.acm.org/citation.cfm?id=2991570

[5] A. Bröring, C. Stasch, and Echterhoff, "OGC Sensor Observation Service Interface Standard, Version 2.0.0 (OGC 12-006)." Open Geospatial Consortium, 2012, p. 163.

[6] A. Cimmino, M. McCool, F. Tavakolizadeh, and K. Toumura, "Web of Things (WoT) Discovery," *W3C Working Draft*, 2021.

[7] Eclipse, "Eclipse vorto-iot toolset for standardized device descriptions, 2016." [Online]. Available: https://www.eclipse.org/vorto/index.html

[8] L. Feigenbaum, G. Todd-Williams, K. Grant-Clark, and E. Torres, "SPARQL 1.1 Protocol," *W3C Recommendation*, 2013.

[9] D. Guinard and V. Trifa, "Towards the web of things: Web mashups for embedded devices," in *Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009), in proceedings of WWW (International World Wide Web Conferences)*, vol. 15, 2009, p. 8.

[10] C. A. Henson, J. Pschorr, A. P. Sheth, and K. Thirunarayan, "SemSOS: Semantic sensor Observation Service," in *2009 International Symposium on Collaborative Technologies and Systems (CTS)*. IEEE Computer Society, 2009, pp. 44–53. [Online]. Available: https://doi.org/10.1109/CTS.2009.5067461

[11] C. Jennings, Z. Shelby, J. Arkko, and A. Keranen, "Media types for sensor markup language (senml)," *Working Draft, IETF Secretariat, Fremont, CA, USA, Tech. Rep. draft-jennings-senml-08. txt*, 2012.

[12] S. M. Jeong, S. Kumar, A. Cimmino, R. García Castro, L. Liquori, M.-A. Peraldi-Frati, E. Scarrone, J. Koss, and F. Bob, "ETSI SmartM2M Technical Report 103717; Study for oneM2M; Discovery and Query specification development," Jun. 2021. [Online]. Available: https://hal.inria.fr/hal-03261080

[13] S. Kaebisch, T. Kamiya, M. McCool, V. Charpenay, and M. Kovatsch, "W3C Web of Things (WoT) Thing Description," *W3C Recommendation*, 2020.

[14] G. Kellog, C. Pierre-Antoine, and D. Longley, "JSON-LD 1.1A JSON-based Serialization for Linked Data," *W3C Recommendation, W3C*, 2020.

[15] M. Kovatsch, R. Matsukura, M. Lagally, T. Kawaguchi, K. Toumura, and K. Kajimoto, "Web of Things (WoT) Architecture," *W3C Recommendation*, 2020.

[16] R. Lea, "Hypercat: an IoT Interoperability Specification," *IoT ecosystem demonstrator interoperability working group*, 2013. [Online]. Available: https://eprints.lancs.ac.uk/id/eprint/69124

[17] L. Liquori, E. Scarrone, M.-A. Peraldi-Frati, S. M. Jeong, A. Cimmino, R. García Castro, J. Koss, A. Q. Khan, S. Kumar, and S. El Khatab, "ETSI SmartM2M Technical Report 103715; Study for oneM2M; Discovery and Query solutions analysis & selection," Jan. 2021. [Online]. Available: https://hal.inria.fr/hal-03115497

[18] S. S. Mathew, Y. Atif, Q. Z. Sheng, and Z. Maamar, "Web of Things: description, discovery and integration," in *2011 International conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing*. IEEE, 2011, pp. 9–15.

[19] D. Nebert, U. Voges, and L. Bigagli, "OGC Catalogue Services 3.0—general model, Version 3.0.0 (OGC 12-168r6)," *OGC Implementation Standard*, 2016. [Online]. Available: http://dx.doi.org/10.25607/OBP-591

[20] M. Nottingham, "Web Linking," ser. Request for Comments, no. 8288. RFC Editor, Oct. 2017. [Online]. Available: https://www.rfc-editor.org/info/rfc8288

[21] C. Perera and A. V. Vasilakos, "A knowledge-based resource discovery for Internet of Things," *Knowledge-Based Systems*, vol. 109, pp. 122–136, 2016.

[22] E. Prud'hommeaux and A. Seaborne, "SPARQL Query Language for RDF," *W3C Recommendation*, 2008. [Online]. Available: https://www.w3.org/TR/rdf-sparql-query/

[23] Z. Shelby, "Constrained RESTful Environments (CoRE) Link Format." RFC Editor, Aug. 2012. [Online]. Available: https://www.rfc-editor.org/info/rfc6690

[24] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," no. 7252, p. 112, Jun. 2014. [Online]. Available: https://www.rfc-editor.org/info/rfc7252

[25] Z. Shelby, S. Krco, and C. Bormann, "CoRE Resource Directory; draft-ietf-core-resource-directory-02," 2014. [Online]. Available: https://datatracker.ietf.org/doc/rfc9176/

[26] J. Swetina, G. Lu, P. Jacobs, F. Ennesser, and J. Song, "Toward a standardized common M2M service layer platform: Introduction to onem2m," *IEEE Wirel. Commun.*, vol. 21, no. 3, pp. 20–26, 2014. [Online]. Available: https://doi.org/10.1109/MWC.2014.6845045

[27] I. Tachmazidis, S. Batsakis, J. Davies, A. Duke, G. Antoniou, and S. S. Clarke, "Optimizing a semantically enriched hypercat-enabled internet of things data hub," in *Proceedings of the 9th International Semantic Sensor Networks Workshop co-located with 17th International Semantic Web Conference (SSN@ISWC)*, vol. 2213. CEUR-WS, 2018, pp. 64–71. [Online]. Available: http://ceur-ws.org/Vol-2213/paper6.pdf

[28] I. Tachmazidis, J. Davies, S. Batsakis, G. Antoniou, A. Duke, and S. S. Clarke, "Hypercat RDF: semantic enrichment for iot," in *Semantic Technology - 6th Joint International Conference (JIST)*, vol. 10055. Springer, 2016, pp. 273–286. [Online]. Available: https://doi.org/10.1007/978-3-319-50112-3_21

[29] F. Tavakolizadeh and S. Devasya, "Thing Directory: Simple and lightweight registry of IoT device metadata," *Journal of Open Source Software*, 2020.

[30] P. Waher and R. Klauck, "Xep-0347: Internet of things-discovery," *Experimental Standard, version 0.1*, vol. 4, 2014.

[31] Y. Zhou, S. De, W. Wang, and K. Moessner, "Search techniques for the Web of Things: A taxonomy and survey," *Sensors*, vol. 16, no. 5, p. 600, 2016.

APPENDIX A: SYNTACTIC AND SEMANTIC QUERIES

Table 3: Syntactic queries defined for experiments

#	JSON Path
Q1	$[?(@.title==%27MyLampThing%27)]
Q2	$[?(@.title=~/.*LampThing/)]
Q3	$.id
Q4	$[?(@.id==%27urn:dev:ops:32473-WoTLamp-1234%27)]
Q5	$

Table 4: Semantic queries defined for experiments

#	SPARQL

Q1	PREFIX td: <https://www.w3.org/2019/wot/td#>
	
	CONSTRUCT {
	?s ?p ?o .
	} WHERE {
	?s td:title "MyLampThing" .
	?s ?p ?o .
	}

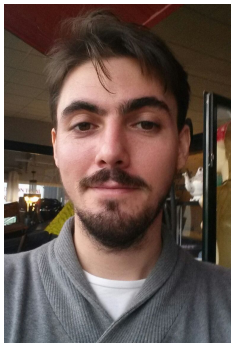
Q2	PREFIX td: <https://www.w3.org/2019/wot/td#>
	
	CONSTRUCT {
	?s ?p ?o .
	} WHERE {
	?s td:title ?t .
	?s ?p ?o .
	FILTER regex(?t, ".*LampThing.*", "i")
	}

Q3	PREFIX td: <https://www.w3.org/2019/wot/td#>
	
	SELECT ?s {
	?s a td:Thing .
	}

Q4	DESCRIBE <urn:dev:ops:32473-WoTLamp-1234>

Q5	CONSTRUCT {
	?s ?p ?o .
	} WHERE {
	?s ?p ?o .
	}

AUTHOR BIOGRAPHIES



Dr. Andrea Cimmino is an Assistant Professor at the Information Systems department at the Universidad Politécnica de Madrid, and researcher at the Ontology Engineering Group. He obtained his Ph.D. in Software Engineering at the Universidad de Sevilla in 2019. Currently he his involved in standardisation bodies and in the program committees of conferences and workshops that are most relevant in his field. His research activities focus on semantic interoperability, data integration, and IoT discovery.

Dr. Raúl García-Castro is Associate Professor at the Computer Science School at Universidad Politécnica de Madrid (UPM), Spain. In 2008 he obtained a Ph.D. in Computer Science and Artificial Intelligence at UPM, which obtained the Ph.D. Extraordinary Award. His research focuses on ontological engineering, semantic interoperability and ontology-based data and application integration. He regularly participates in standardisation bodies and in the program committees of the conferences and workshops that are most relevant in his field, having also organised several international conferences and workshops.