Pers Ubiquit Comput (2017) 21:475–487
DOI 10.1007/s00779-017-1010-8

ORIGINAL ARTICLE

IoT-Lite: a lightweight semantic model for the internet of things and its use with dynamic semantics

Maria Bermudez-Edo[1] · Tarek Elsaleh[2] · Payam Barnaghi[2] · Kerry Taylor[2,3]

Received: 8 October 2016 / Accepted: 30 December 2016 / Published online: 22 February 2017

Abstract Over the past few years, the semantics community has developed several ontologies to describe concepts and relationships for internet of things (IoT) applications. A key problem is that most of the IoT-related semantic descriptions are not as widely adopted as expected. One of the main concerns of users and developers is that semantic techniques increase the complexity and processing time, and therefore, they are unsuitable for dynamic and responsive environments such as the IoT. To address this concern, we propose IoT-Lite, an instantiation of the semantic sensor network ontology to describe key IoT concepts allowing interoperability and discovery of sensory data in heterogeneous IoT platforms by a lightweight semantics. We propose 10 rules for good and scalable semantic model design and follow them to create IoT-Lite. We also demonstrate the scalability of IoT-Lite by providing some experimental analysis and assess IoT-Lite against another solution in terms of round trip time performance for query-response times. We have linked IoT-Lite with stream annotation ontology, to allow queries over stream data annotations, and we have also added dynamic semantics in the form of MathML annotations to IoT-Lite. Dynamic semantics allows the annotation of spatio-temporal values, reducing storage requirements and therefore the response time for queries. Dynamic semantics stores mathematical formulas to recover estimated values when actual values are missing.

Keywords Internet of things · Semantics · Linked sensor data · Knowledge management · Dynamic semantics

✉ Maria Bermudez-Edo
mbe@ugr.es

Tarek Elsaleh
t.elsaleh@surrey.ac.uk

Payam Barnaghi
p.barnaghi@surrey.ac.uk

Kerry Taylor
Kerry.Taylor@anu.edu.au

[1] Software Engineering Department, University of Granada, Granada, Spain

[2] Institute for Communication Systems, University of Surrey, Guildford, United Kingdom

[3] Research School of Computer Science, Australian National University, Canberra, Australia

1 Introduction

With the growing development of machine-to-machine (M2M) communications and IoT deployments, interoperability between different platforms has become a key issue in creating large-scale IoT frameworks. Semantic technologies suggest a suitable approach for interoperability by sharing common vocabularies, and also enabling interoperable representation of inferred data. IoT testbed providers have recently started to add semantics to their frameworks allowing the creation of the semantic sensor web (SSW), which is an extension of the current Web in which information is given well-defined meaning, enabling M2M communications and interactions between objects, devices and people [26].

Semantics often models domain concepts in great detail. Although they can be applied for querying almost anything about objects, these complex models are often difficult to implement and use, especially by non-experts. They demand considerable processing resources, and therefore, they are considered unsuitable for constrained environments. Instead, IoT models should be designed for the

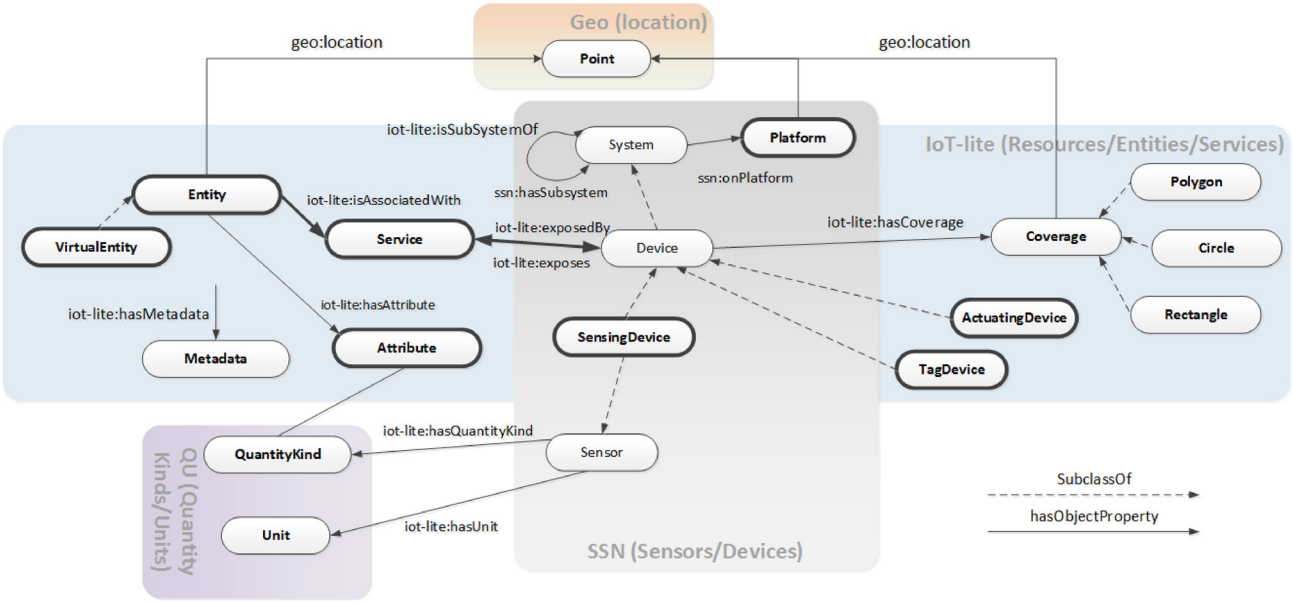

Fig. 1 An overview of the proposed semantic model, IoT-Lite

constraints and dynamicity of IoT environments, especially recognising the new trend towards integrating semantic processing on constrained devices such as M2M gateways or smartphones. At the same time, they need to model the relationships and concepts that represent and allow interoperability between IoT entities. Therefore, expressiveness versus complexity is a challenge. One of the key issues in heterogeneous IoT ecosystems is accessing sensor data from different systems. Enabling a lightweight description of sensors to efficiently manage annotation and discovery of sensor data is essential.

It is important to note that semantic models are not end-products. They are normally only part of a solution and should be transparent to the end user. Semantic annotation models should be offered with effective methods, APIs and tools to process the semantics in order to extract actionable information from raw data. Query methods, machine learning, reasoning and data analysis techniques should be able to effectively use these semantics. Semantic modelling is only the initial part of the whole design, and it has to take into account how the models will be used; how the annotated data will be indexed and queried with real-time data; and how to make the data publication suitable for constrained environments and large-scale deployments when applications often require low latency and processing time.

We propose IoT-Lite, a lightweight semantic model which is an instantiation of the semantic sensor network (SSN) ontology [8] (see Fig. 1). IoT-Lite is the outcome of a research effort that focuses on loosely-coupled discovery of real-time sensor data and seeks for the minimum concepts and relationships that can provide answers to most of the end user queries. We have focused on the typical queries for accessing the data in the IoT based on our experience in the challenge of analysing data for obtaining meaningful information for end-users. We find that we do not need full descriptions and complex relationships to satisfy user queries. Some of the most commonly used semantic models on the Web are simple models, such as friend of a friend, (FOAF).[1] Their simplicity encourages faster adoption by end-users, as they do not imply complex annotations and they do not require complex processing methods. Simpler models can also support providing faster responses to queries.

In this paper, we also propose guidelines for developing scalable and reusable semantic models in the IoT. These guidelines leverage conventions followed by some semantic modelling designers, such as the linked data approach.

IoT-Lite does not intend to be a full ontology for the IoT. Our aim is to create a core lightweight ontology that allows relatively fast annotation and processing time. IoT-Lite can be a core part of a semantic model in which, depending on the applications, different semantic modules can be added to provide additional domain and application-specific concepts and relationships. In this sense, we have linked IoT-Lite to stream annotation ontology (SAO) [16], in order to allow the annotation of aggregated data streams, which follows the philosophy of IoT-Lite in the sense of lightweight ontology and fast response time to queries.

Finally, we propose the use of dynamic semantics and demonstrate a use-case in the form of MathML annotations that can be used together with IoT-Lite. Dynamic semantics can be used to represent formulas that extrapolate missing values. By following the IoT-Lite approach,

[1] http://www.foaf-project.org/.

dynamic semantics reduces the size of the triple store and offers fast response times to queries. Unlike other solutions such as the use of RESTful servers to extrapolate missing values, IoT-Lite stores all the information about the stream data together in one place, the triple store.

The remainder of the paper is organised as follows. Section 2 describes the related work. Section 3 introduces the 10 rules for good and scalable semantic model design and presents the proposed model, IoT-Lite, for representation of IoT elements. Section 4 provides a use-case scenario that illustrates the semantic annotation of a sensor in our model. Section 5 details an evaluation of the proposed model against a more detailed model. Section 6 shows an example of the use of IoT-Lite together with SAO. Section 7 introduces dynamic semantics and an example with IoT-Lite. Finally, Sect. 8 concludes the paper and describes the future work.

2 Related work

There are several semantic descriptions designed for the IoT domain. The SSN ontology [8] is one of the most significant and widespread models to describe sensors and IoT-related concepts.

The SSN ontology provides concepts describing sensors, such as outputs, observation value, feature observed, observation time, accuracy, precision, deployment configuration, method of sensing, system structure, sensing platforms and feature of interest. However, it is a detailed description, containing concepts and properties that enable flexible descriptions over a very wide range of applications, but including non-essential components for many use cases that can make the ontology heavy to query and process if it is used as it is.

The IoT-A model[2] and IoT.est [30] are some of the many projects that extend the SSN ontology to represent other IoT-related concepts such as services and objects in addition to sensor devices. IoT-A provides an architectural base for further IoT projects (see Fig. 2). The only implementation of a purely IoT-A semantic model known by the authors is described in [11]. The IoT-A model is overly complex for fast user adaptation and responsive environments. The IoT.est model extends the IoT-A model with extended service and test concepts.

The open geospatial consortium (OGC), through its sensor web enablement (SWE) group [7], has developed a set of standards to describe sensors and their data. For example, SensorML,[3] which is an XML language to describe the sensing process, and observations and measurements (O&M), which is a UML model (with an XML form) and from which the observation concept in SSN was derived. While SensorML provides important syntactic descriptions using XML, it lacks the expressibility provided by ontology languages such as OWL. SemSOS [13] has mapped the XML tags of O&M into OWL concepts. However, it represents only observations and not other IoT-related concepts. OMLite is a new ontology that also re-states O&M as an ontology, but likewise misses IoT concepts [9].

One of the ongoing works is OneM2M. OneM2M has published a report for home automation and describes concepts and relationships [23]. Another current initiative is the spatial data on the Web Working Group,[4] a joint effort between the world wide web consortium (W3C) and the open geospatial consortium (OGC) that aims to standardise key ontologies for spatial, temporal and sensor data on the web [28]. Several projects also work on semantic descriptions for the IoT, such as FED4FIRE[5] that currently has a semantic model focused on communications, VITAL[6] for smart cities, CityPulse[7] with more focus on data [18] and OpenIoT,[8] which is an extension of SSN.

Performance of ontologies for large data sets has been addressed by different methods, such as by redesigning the data storage model and leveraging expected query patterns [27]. Our proposed IoT-Lite ontology extends previous works and can be used in combination with other techniques for querying performance improvements, such as the dynamic semantics we propose in Sect. 7.

To summarise, existing published IoT ontologies are either complex or domain-specific for sub-domains of IoT. The creation of a lightweight ontology that allows interoperability and discovery of sensory data in heterogeneous platforms with low complexity and processing time is still an open issue.

The majority of current semantic annotation techniques and semantic description frameworks are static and are based on the assumption that the stored data will not change over time. Researchers search for solutions for dynamic data outside the semantic annotations, such as by using RESTful servers [24] that access real-time data or infer missing values (e.g. [15, 20]). However, these solutions need to access several servers and are not suitable for low-connectivity networks or high-frequency queries. To the best of our knowledge, none of the solutions includes the dynamicity inside the semantic descriptions. Other attempts to handle dynamic data in semantics are based on reasoning. For example, Baader et al., have introduced the temporised

[2] http://www.iot-a.eu/.

[3] http://www.opengeospatial.org/standards/sensorml.

[4] http://www.w3.org/2015/spatial/.

[5] http://www.fed4fire.eu/.

[6] http://vital-iot.eu/.

[7] http://www.ict-citypulse.eu/.

[8] http://www.openiot.eu/.

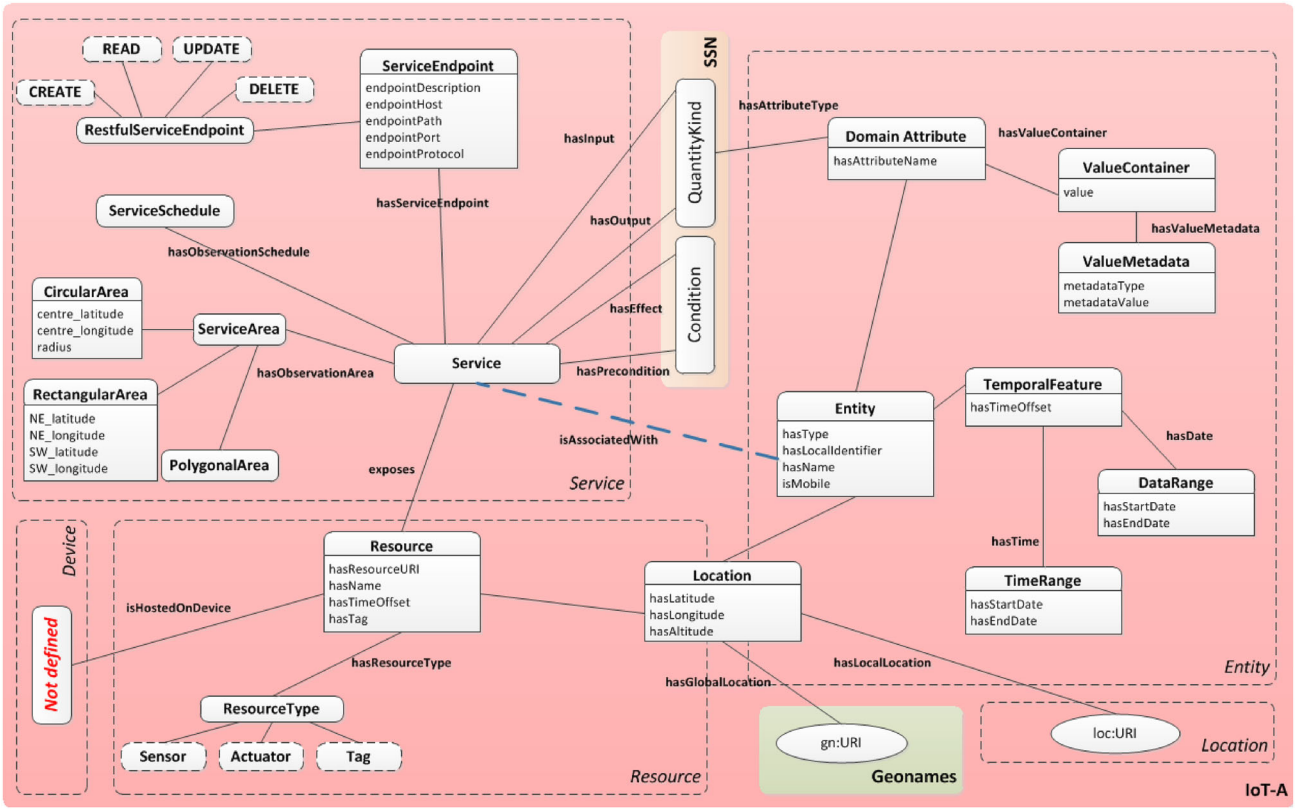

Fig. 2 An overview of the IoT-A ontology

description logic, which provides semantic reasoning with a temporal component [1]. Lopez et al. [19] have implemented static mathematical formulas, as rules, inside the semantic descriptions, but these are not dynamic as our solution. These formulas are static, they do not have spatio-temporal components and their execution is done with a low frequency. They are also more restricted than ours, as they do not have the flexibility of MathML in handling any mathematical description. Furthermore, most of the current reasoning engines are not efficient in dealing with high volumes of queries with real-time requirements (see for example [3, 5]).

Our dynamic semantics solution differs from the previous works in dynamic semantic annotation, because the dynamicity is annotated inside the semantic descriptions as a MathML formula. This formula can be executed outside the semantic description, avoiding the processing time of semantic reasoners, but at the same time avoiding the need to access the data or the abstraction of the data in different servers.

3 IoT-Lite: IoT modelling and semantic annotation

While most of the semantic models tend to describe the concepts in great detail and represent various links in IoT systems, we represent only the most used concepts for data analytics in IoT applications, such as sensory data, location and type. See Fig. 1 for the model and Fig. 3 for an example of an annotated sensor. This paves the way for creating scalable responsive systems and reduces memory and computational cost of query processing in large-scale IoT applications.

In 2003, W3C published a list of sample "Good Ontologies" following specific good practices.[9] The goodness of the ontologies was scored based on five aspects: fully documented; dereferenceable; used by independent data providers; possibly supported by existing tools; and in use by two independent datasets.

IoT-Lite addresses these aspects to create a reusable model. We have published the ontology with a web page that fully documents the ontology (aspect 1) with a permanent link,[10] and all the concepts in the ontology are described by a dereferenceable URI (aspect 2) [4]. The annotations are applied to IoT testbeds, University of Surrey SmartCampus [21] and SmartSantander [25]. They are planned to be used by other independent platforms in the open calls of the H2020 project FIESTA-IoT[11] (aspects 3 and 5). We plan to develop annotation and validation

[9] http://www.w3.org/wiki/Good_Ontologies.

[10] http://www.purl.oclc.org/NET/UNIS/fiware/iot-lite.

[11] http://fiesta-iot.eu.

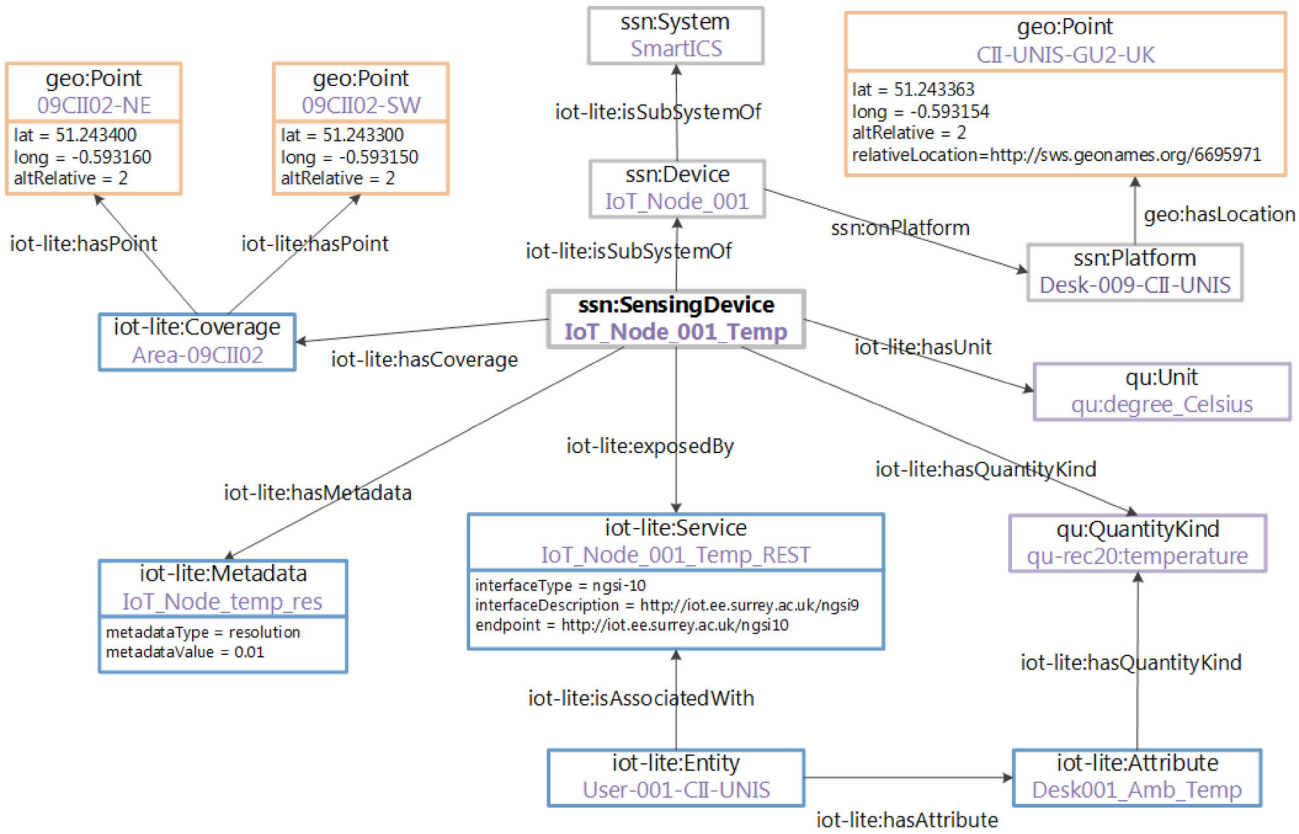

Fig. 3 An example of a sensor annotated with the proposed IoT-Lite ontology

tools for IoT-Lite, by extending our SAOPY annotation tool[12] [16] and the SSN validator tool[13] (aspect 4).

Although the above aspects are essential to create interoperable and reusable ontologies, they are not enough to cover scalability, dynamicity and user adoption issues. We propose a set of guidelines for developing scalable ontologies.

1. Design for large scale.
2. Think of who will use the semantics and design for their needs.
3. Provide means to update and change the semantic annotations.
4. Create tools for validation and interoperability testing.
5. Create taxonomies and vocabularies.
6. Reuse existing models.
7. Link data and descriptions to other existing resources.
8. Define rules and/or best practices for providing the values for each property.
9. Keep it simple.
10. Create effective methods, tools and APIs to handle and process the semantics.

In the design of IoT-Lite, we have followed these rules. We have designed a lightweight ontology considering the scalability (following rule 1) and will provide tools for annotation and validation (rule 3 and 4), as well as APIs and using existing tools for querying and information processing (rule 10) as we mentioned previously. Semantics are only one part of the solution and often not the end-product. Query methods, machine learning, reasoning and data analysis techniques and methods should be able to effectively use these semantics.

We have designed IoT-Lite (see Fig. 1) with a clear purpose of defining only the most used terms when searching for IoT concepts in the context of data analytics. We studied the most common uses of IoT ontologies (following rule 2) based on our experience with other IoT ontologies used by applications for data analytics. For example, an application that provides the temperature on the move will query the ontology for the temperature sensor service endpoint at each particular location. The ontology needs the concept of sensor, the quality it measures (temperature) coverage and endpoint. Other concepts are irrelevant in that query. The ontology needs also to

[12] https://github.com/CityPulse/SAOPY.

[13] http://iot.ee.surrey.ac.uk/SSNValidation/.

have these concepts easily accessible, avoiding deep and distantly-connected terms of the ontology that need complex queries to retrieve the desired results. Therefore, the simplicity of the ontology is essential (rule 9). The most widely used semantic descriptions on the Web are simple ones such as FOAF.

Another important aspect of semantic models is the interoperability. In the design of IoT-Lite, we followed the linked data guidelines[14]. Our ontology is linked with other ontologies (rule 6 and 7). We chose well known and widely used ontologies, expecting their publications to be stable (e.g. SWEET and SSN). We avoid links to uncommonly used ontologies in order to prevent inconsistencies in case of unexpected deletion of the linked ontologies. In the context of interoperability, it is also important to use the same vocabulary to be able to share and combine data from different sources. For that reason, we have created a taxonomy of quantity kinds and units which is published on the ontology webpage and is a compilation of terms used in well known ontologies such as qu[15] and qudt[16] (rule 5 and 7). IoT-Lite is published with a webpage which fully explains the terms used and provides examples (rule 8). This allows reuse and linking with other ontologies.

With the above prerequisites, we have created an extension of SSN, which is considered the de facto standard of sensor networks ontologies. Furthermore, it is currently on-track for formal standardisation through the OGC and the W3C [28]. SSN is not designed to be necessarily used as it is in full form; it is a template to be extended and instantiated. We have customised SSN to make a lightweight ontology with the main concepts being the three well-accepted items in the classification of IoT entities [29]: Entities or objects; resources or devices; and services, namely `iot-lite:Object`, `ssn:Device` and `iot-lite:Service`. Figure 1 shows an overview of the proposed information model. These three concepts are the core concepts of the ontology and are necessary in any ontology describing IoT.

The relationships between these three concepts are also well known [10, 12], that is, an object (or entity) `iot-lite:Object` has an attribute `iot-lite:Attribute` which is associated with a device (or resource) `iot-lite:Device`, which is exposed by a service `iot-lite:Service`. We built the rest of the ontology around these three main concepts adding the necessary concepts and relationships to provide responses to the standard queries. The objects can be moving objects and therefore the relationship, or association, between the objects and the devices are dynamic. For example, a bicycle can be associated with a pollution sensor in one street, but when the bicycle moves to another street, it will be associated with a different pollution sensor.

To allow the queries to be lighter, we have linked most of the concepts of the ontology under one main class (Device) and leave the other two classes lighter. We have spotted at least three main classes of Devices (`ssn:Sensor`, `iot-lite:Actuator`, `iot-lite:Tag`) that we need to separate due to the differences that applications can query for. For example, an application that needs to know the temperature will query for sensors, while if the application needs to switch on the lights, it will query for actuators. `ssn:SensingDevice` is directly linked via properties or via inheritance of the relevant properties to the concepts `qu:QuantityKind`, `qu:Units` and `iot-lite:Coverage`. Therefore, we need only three triples to link each sensing device with these concepts (e.g. Sensor1 has QuantityKind temperature).

In order to allow a common vocabulary to interoperate between different systems, we need a taxonomy to describe the measurements of the devices in terms of the quantity kinds and units, such as temperature and degrees celsius. We have created this taxonomy using individuals from well-know ontologies, such as qu-rec20[17] and qudt.[18]

The spatial dimension of the ontology is addressed with the geo ontology[19] based on WGS84 location coordinates.[20] This simple ontology is widely used, and there are some available tools for discovery whether a point belongs to an area, (circle, rectangle or polygon), and extensions to SPARQL to deal with geolocations, such as the OGC standard GeoSPARQL. We have added relative locations to these geolocations to annotate locations such as a building or a floor in indoor scenarios, where the geolocation is less intuitive. The relative location also supports linking to resources such as GeoNames[21] that are publicly available as part of the Linked Open Data cloud.[22]

4 Use-case

In this section, we exemplify the use of the proposed information model, IoT-Lite, using sensor information from the Surrey testbed [22] developed within the EU FP7 project Smart Santander.[23] The testbed consists of 200 IoT

[14] http://linkeddata.org/.

[15] http://www.w3.org/2005/Incubator/ssn/ssnx/qu/qu-rec20.html.

[16] http://www.qudt.org/qudt/owl/1.0.0/quantity/.

[17] http://purl.org/NET/ssnx/qu/qu-rec20.

[18] http://www.qudt.org/qudt/owl/1.0.0/quantity.

[19] http://www.w3.org/2003/01/geo/wgs84_pos.

[20] http://confluence.qps.nl/pages/viewpage.action?pageId=29855173.

[21] http://www.geonames.org/.

[22] http://lod-cloud.net/.

[23] http://www.smartsantander.eu/.

nodes/devices provided with 6 sensors each that measure temperature, sound, vibration, light, presence and energy consumption.

Figure 3 illustrates a sample describing the output of one of the temperature sensors in the testbed using the IoT-Lite ontology. This sensor is associated with the temperature of a room. In this example, it can be seen that a table located in Room CII01 has an attribute, temperature, which is associated with the temperature sensor located in the same room. The temperature sensor has a coverage that covers the area of the room (rectangle), defined by two points in the diagonal corners; measures the temperature with degrees celsius and a resolution of 0.01; and is exposed by a service with endpoint http://surrey.ac.uk/sensor/roomCII01. We have used the geolocation to annotate the latitude and the longitude coordinates. However, we have also annotated the relative altitude as floor 1 for better human understanding. Listing 1 is an excerpt of the same temperature sensor annotation in a turtle format.

```
@prefix qu: <http://purl.org/NET/ssnx/qu/qu#> .
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix ssn: <http://www.w3.org/2005/Incubator/ssn/ssnx/ssn#> .
@prefix iot-lite:<http://purl.oclc.org/NET/UNIS/iot-lite/iot-lite#>

:temperatureSensorRoom13CII01 rdf:type owl:NamedIndividual ,
ssn:Sensor ;

iot-lite:type "SensorTelosB"^^xsd:string ;
iot-lite:id "telosB-001"^^xsd:string ;
geo:hasLocation :locationRoom13CII01 ;
iot-lite:exposedBy :ngsi10SensorRoom13CII01 ;
iot-lite:hasMetadata :resolution1024 ;
iot-lite:hasUnit qu:degree_Celsius ;
iot-lite:hasQuantityKind qu:temperature .

iot-lite:hasCoverage :areaRoom13CII01 ;

iot-lite:tableRoom13CII01 rdf:type iot-lite:Object ,
owl:NamedIndividual ;

iot-lite:description "http://Room13CII01/Tabl"^^xsd:anyURI ;
iot-lite:hasAttribute iot-lite:temperaturTableRoom13CII01 ;
geo:hasLocation :locationRoom13CII01 .

iot-lite:temperatureTableRoom12CII01 rdf:type iot-lite:Attribute ,
owl:NamedIndividual ;

iot-lite:isAssociatedWith :temperatureSensorRoom13CII01 .

:areaRoom13CII01 rdf:type iot-lite:Rectangle ,
owl:NamedIndividual ;

iot-lite:hasPoint :NEcornrRoom13CII01 , :SWcornrRoom13CII01 .

:NEcornerRoom13CII01 rdf:type owl:NamedIndividual ,
geo:Point ;

geo:long "-0.59316"^^xsd:float ;
iot-lite:altRelative "1stFloor"^^xsd:string ;
geo:lat "51.2434"^^xsd:float .

:SWcornerRoom13CII01 rdf:type owl:NamedIndividual ,
geo:Point ;

geo:long "-0.59315"^^xsd:float ;
iot-lite:altRelative "1stFloor"^^xsd:string ;
geo:lat "51.2433"^^xsd:float .

:locationRoom13CII01 rdf:type owl:NamedIndividual ,
geo:Point ;

geo:long "-0.593154"^^xsd:float ;
iot-lite:altRelative "1stFloor"^^xsd:string ;
geo:lat "51.243362"^^xsd:float .

:ngsi10SensorRoom13CII01 rdf:type iot-lite:Service ,
owl:NamedIndividual ;

iot-lite:endpoint "http://meassur/rom13CII01"^^xsd:anyURI ;
iot-lite:description "http://meassur/room13CII01"^^xsd:anyURI ;
iot-lite:serviceType "ngsi-10"^^xsd:string ;

:resolution1024 rdf:type iot-lite:Metadata ,
owl:NamedIndividual ;

iot-lite:value "0.01"^^xsd:float ;
iot-lite:metadataType "resolution"^^xsd:string .
```

Listing 1: An excerpt from a sensor annotation based on IoT-Lite Ontology.

5 Evaluation

In order to validate the scalability and applicability of IoT-Lite, we performed some experiments using sensory data from the University of Surrey's SmartCampus testbed. A web application developed in Java was used to annotate the ontology individuals that represent the sensing devices and to store them in a set of Jena TDB triple stores,[24] one for each dataset. We used a personal computer (PC) running Windows 7 (x64) operating system with a processor Intel(R) Core(TM) i5-3470 CPU @ 3.20GHz 8GB RAM to act as a server that hosts the web application. We sent remote queries from a different PC located in another subnet. The aim of this experiment was to measure the response time of a common query. With IoT-Lite, a common query is defined as in Listing 2 as a query asking for the endpoint of the services that provide the temperature in a particular area. As can be seen, the query is simple. It contains just six triples due to the shallow depth of the IoT-Lite ontology. Likewise the same query in IoT-A contains ten triples (Listing 3).

```
SELECT ?sens ?endp
       WHERE {
          ?sensDev iot-lite:hasQuantityKind qu-rec20:temperature;
          iot-lite:isExposedBy ?serv;
          iot-lite:hasCoverage ?cover.
          ?cover iot-lite:hasPoint ?point.
          ?point iot-lite:RelativeLocation "Desk2".
          ?serv iot-lite:endpoint ?endp.
       }
```

Listing 2: Query performed in the experiments in IoT-Lite ontology.

```
SELECT ?iotService ?endpHost ?endpPort ?endpPath ?endpProt
       WHERE {
          ?iotService serv:hasOutput qu-rec20:temperature.
          ?iotService serv:hasServiceEndpoint ?endp.
          ?endp serv:endpointHost ?endpHost.
          ?endp serv:endpointPort ?endpPort.
          ?endp serv:endpointPath ?endpPath.
          ?endp serv:endpointProtocol ?endpProt.
          ?iotService serv:exposes ?res.
          ?res res:hasResourceType ssn:Sensor.
          ?res res:hasLocation ?loc.
          ?loc res:hasGlobalLocation ""GU1""
       }
```

Listing 3: Query performed in the experiments in IoT-A ontology.

We performed this query over different datasets. For that purpose, we created four datasets containing 200, 1000, 10,000 and 100,000 sensors each. The IoT-Lite ontology contains 116 triples by itself. When annotating sensors, each new sensor needs just six triples, and in total the number of triples in each data set is shown in Table 1.

To compare the ontology against other solutions, we performed the same experiments with IoT-A, another instantiation of SSN aiming to define the architecture of IoT. We chose IoT-A because we have used the IoT-A ontology in one of our components, a discovery element for IoT entities. With this ontology, we experienced some of the problems mentioned in the introduction, and this motivated us to develop IoT-Lite to replace IoT-A in the discovery component. Figure 2 shows IoT-A. We queried IoT-A with a similar query to that for IoT-Lite, but in this case we needed ten triples to obtain the same results, i.e. the endpoints of services that provide the temperature in a particular area. The IoT-A ontology contains 346 triples by itself. The total number of triples of each data set is also shown in Table 1.

In order to avoid false perceptions of the round time trip (RTT) due to jitter, we sent the query ten times to each dataset. Figure 4 shows the boxplot results of these 10 queries for each dataset. We can see that the RTT of the query/response is acceptable for every dataset in IoT-Lite. Even when the dataset contains 100,000 individuals the mean of the RRT is below 200 ms. We can also see that the time of the RTT is less in IoT-Lite than in IoT-A in all the cases, and particularly in large datasets, such as 100,000 sensors, the time of IoT-A is more than twice the time of

[24] https://jena.apache.org/documentation/tdb/.

Table 1 Number of triples in each dataset

Datasets: number of sensors	200	1000	10,000	100,000
Number of triples in IoT-Lite	1486	6926	68,126	680,126
Number of triples in IoT-A	1866	7946	76,346	760,346

IoT-Lite. IoT-Lite performs better than Iot-A for large-scale annotations of sensors.

6 Extending IoT-Lite with data aggregation

When dealing with IoT applications, one of the important issues to take into account is the immense amount of data generated. Most applications could work properly with less data, efficiently aggregated. With the same aim as IoT-Lite, the SAO ontology (stream annotation ontology) was created in order to deal with huge amount of IoT data in a efficient manner [18]. The SAO ontology provides annotation means to represent aggregated data in a lightweight ontology [16].

To demonstrate both the extensibility of IoT-Lite and the use of our ontology for IoT data analytics, we have linked it with the SAO ontology as shown in Fig. 5. In order to show the connections between both ontologies clearly, we have represented only the main classes of both ontologies in Fig. 5. We took advantage of the common base in SSN of both, SAO and IoT-Lite. SAO is linked with SSN through the class `ssn:Sensor` which is a superclass of `ssn:SensingDevice`. Therefore, the link is straightforward via the property `ssn:observedBy`. With this connexion, IoT-Lite allows to access the raw data via an endpoint, data can be in any format, semantic or not, as shown previously in Listing 1; IoT-Lite also allows access to aggregated semantic data. Listing 4 shows an example in turtle of the same sensor shown in Listing 1, but this time annotated with SAO ontology and with a sampling frequency of 1 h. In this example, we have sampled the data taking one sample every hour, although SAO permits various other aggregation algorithms.

```
@prefix qu: <http://purl.org/NET/ssnx/qu/qu#> .
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix ssn: <http://www.w3.org/2005/Incubator/ssn/ssnx/ssn#> .
@prefix iot-lite:<http://purl.oclc.org/NET/UNIS/iot-lite/iot-lite#>
@prefix sao: <http://example.com#> .
@prefix tl: <http://purl.org/NET/c4dm/timeline.owl#> .

:temperatureSensorRoom13CII01 rdf:type owl:NamedIndividual ,
ssn:Sensor ;

iot-lite:type "SensorTelosB"^^xsd:string ;
iot-lite:id "telosB-001"^^xsd:string ;
geo:hasLocation :locationRoom13CII01 ;
iot-lite:hasUnit qu:degree_Celsius ;
iot-lite:hasQuantityKind qu:temperature .
iot-lite:hasCoverage :areaRoom13CII01 ;

iot-lite:tableRoom13CII01 rdf:type iot-lite:Object ,
owl:NamedIndividual ;

iot-lite:description "http://Room13CII01/Tabl"^^xsd:anyURI ;
iot-lite:hasAttribute iot-lite:temperaturTableRoom13CII01 ;
geo:hasLocation :locationRoom13CII01 .

iot-lite:temperatureTableRoom12CII01 rdf:type iot-lite:Attribute ,
owl:NamedIndividual ;

iot-lite:isAssociatedWith :temperatureSensorRoom13CII01 .

:areaRoom13CII01 rdf:type iot-lite:Rectangle ,

owl:NamedIndividual ;
iot-lite:hasPoint :NEcornrRoom13CII01 , :SWcornrRoom13CII01 .

:NEcornerRoom13CII01 rdf:type owl:NamedIndividual ,

geo:Point ;
geo:long "-0.59316"^^xsd:float ;
iot-lite:altRelative "1stFloor"^^xsd:string ;
geo:lat "51.2434"^^xsd:float .

:SWcornerRoom13CII01 rdf:type owl:NamedIndividual ,

geo:Point ;
geo:long "-0.59315"^^xsd:float ;
iot-lite:altRelative "1stFloor"^^xsd:string ;
geo:lat "51.2433"^^xsd:float .

:locationRoom13CII01 rdf:type owl:NamedIndividual ,

geo:Point ;
geo:long "-0.593154"^^xsd:float ;
iot-lite:altRelative "1stFloor"^^xsd:string ;
geo:lat "51.243362"^^xsd:float .

:temperatureSensorRoom13CII01Observation-001 rdf:type owl:NamedIndividual ,
sao:Point ;

sao:value "24.0"^^xsd:double ;
sao:time [   a tl:Instant         ;
tl:at "2016-09-02T10:00:00"^^xsd:dateTime   ;
tl:duration "PT1H"^^xsd:duration
] ;
ssn:observedBy :temperatureSensorRoom13CII01 .
```

Listing 4: An excerpt from a sensor annotation based on IoT-Lite Ontology linked to the aggregated data coming out of the sensor with an aggregation algorithm

7 Extrapolating data via dynamic semantics

Data stored in a triple store can have a coarser granularity than needed by one application. The coarse granularity of the data may be due to constraints in the sensors, such as on their capacity to store data or the frequency to read or send

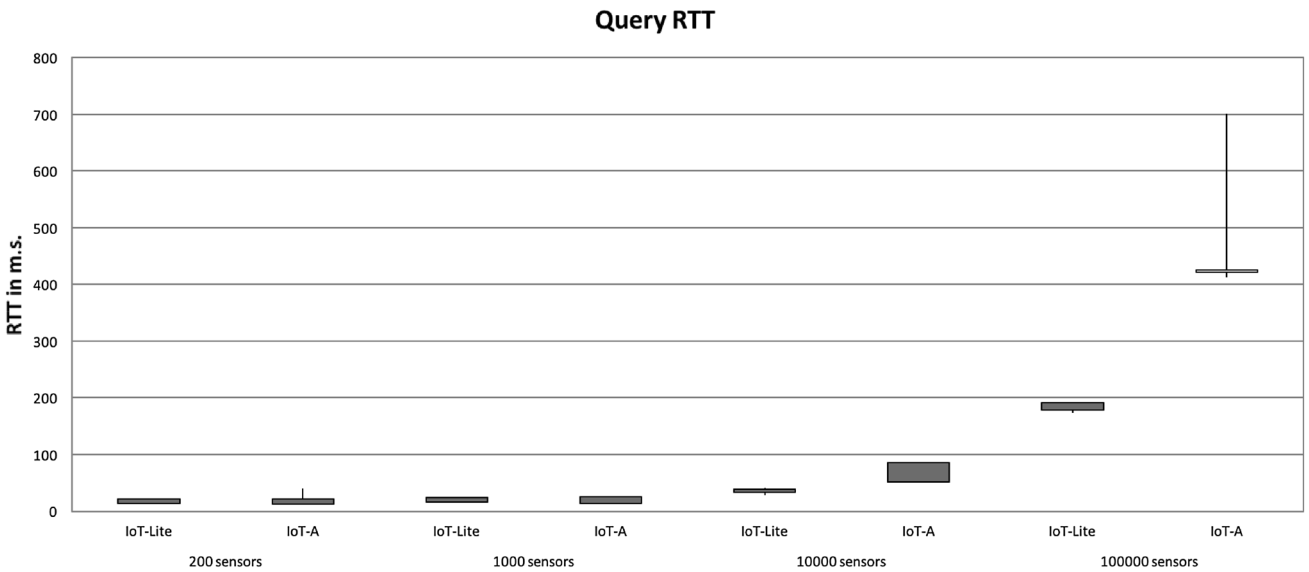

Fig. 4 Boxplot of the round time trip (RTT) of the queries required to retrieve the endpoint of a temperature sensor in a certain location depending on the size of the triple store with both ontologies IoT-Lite and IoT-A

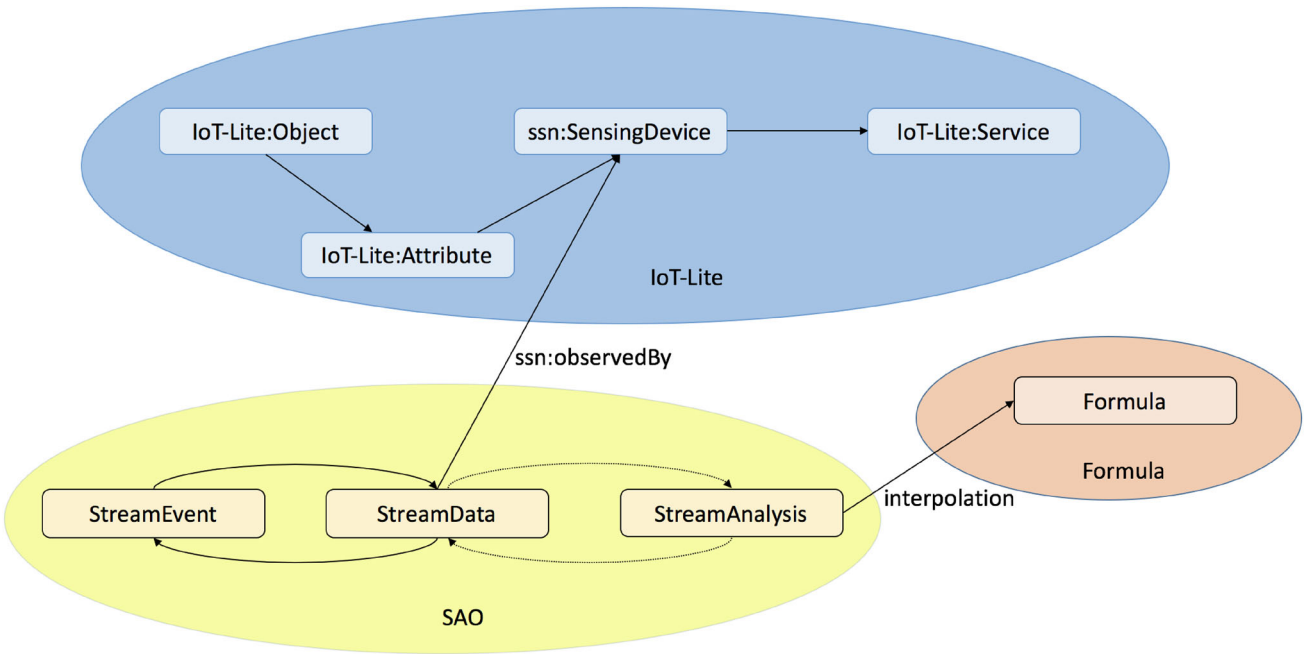

Fig. 5 IoT-Lite linked with SAO ontology for data aggregation and Formulas ontology for recovery or interpolation of data

data, or constraints in the network such as communication bandwidth or storage [2]. Some times the raw data has finer granularity and a posteriori, either sampling or aggregation algorithms are applied in order to reduce the amount of data stored. In other cases, the sensors only provide coarse granularity. In all these cases, we can interpolate the data by applying any interpolation or recovery algorithm that infers the missing values. Thus, the application can have coarser granularity than the triples.

In semantics, annotations are typically static, i.e. they store static values. Dynamic streams can be annotated with semantics, but in a static manner, i.e. annotating the stream values as they are produced, but once the values are annotated they become static values. In order to have dynamic annotated values, we have developed the dynamic semantics. Dynamic semantics aims at having more flexibility in the ontologies. Our first approach in this sense is to store formulas linked to the data values that allow users to

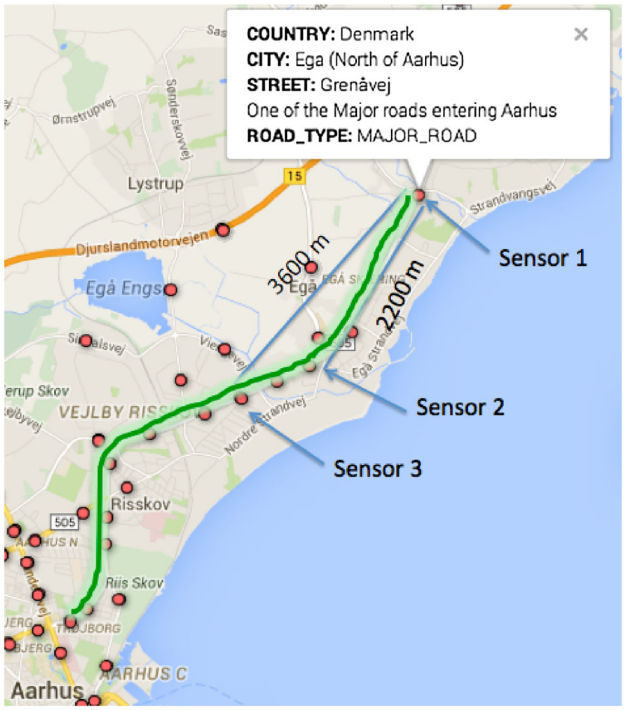

Fig. 6 One of the main roads that connect the city centre of Aarhus with the surrounding towns. The *dots* represent the location of the traffic sensors. We use sensor 1 and sensor 3 to infer the prediction model and sensor 2 for testing purposes

derive dynamic values out of static values given the right parameters and formula. For example, if we have the measurements of the traffic in a city coming from sensors set up in different points of the city and stored only every 3 h, and we have a good simulation model for traffic which we can store inside the triple store, we can interpolate the stored values to obtain a traffic value at any place and any sparse time in the city on demand only by reading data from the triple store.

The dynamic semantics use-case discussed here uses MathML[25] to store the formulas as data property literals. MathML is a W3C recommendation for a mark-up language to describe mathematical expressions, and it is widely used. This solution keeps all the information in the same description thus avoiding access to different servers to access calculations. Furthermore, there exist several readers and converters that express MathML expressions in different languages, such as Java, Python or C++ and vice-versa. The method could be used not only for interpolation, but also for forecasting into future extrapolation.

To demonstrate the usability of dynamic semantics, we present a case study in a smart city. We have performed an experiment with the public traffic data[26] obtained from the city of Aarhus in Denmark. The dataset consists of traffic data measured every 5 min using 135 sensors located in different parts of the city. The data are organised in pairs of sensors providing information regarding the geographical location of sensors, time-stamp and traffic intensity such as average speed and vehicle count. Figure 6 shows the location of some of the sensors in Aarhus on a Google Map.

In particular, we will study the patterns of vehicular traffic and focus on traffic data in the early hours of business days. We will store this model as a formula in our ontology. The first step in our experiment is to create the prediction model for traffic patterns. This includes two linear interpolation models, one for the spatial dimension and the other for the temporal dimension of the prediction model.

To infer the model, we captured the traffic data from Aarhus for a period of 2 months (August–September 2014). The dataset is available online on the EU FP7 CityPulse Project datasets Web page.[27] We measure the traffic (number of vehicles) entering the city of Aarhus through a main road. We took the data from two sensors separated by 3600 m (sensor1 and sensor 3 in Fig. 6), on working days within a 3 h time period where the traffic gradually increases in the early hours of the morning before working hours. These data (pattern) are represented with the formula:

$$\begin{aligned} V_x = {} & \text{LastVehicleCount} \\ & + 0.126603432701 \times (\text{currentTimeInMinutes} - 180) \\ & + 0.000765329644997 \times (\text{CurrentLocation}) \end{aligned}$$

This formula is described based on spatial and temporal dependencies. We include this mathematical expression in the formula module of the semantic model (see Fig. 5). Later on, when a user accesses the traffic data, the formula from the semantic descriptions can provide the user with an estimation of the current value of the traffic data at an specific location and also provide an estimate value for other adjunct locations (assuming a model has been constructed to generate those values).

In order to write and read the semantic annotations, we have used the libSBML library [6] that performs the translations between Python code and MathML. LibSBML is a specific library for writing and manipulating the Systems Biology Mark-up Language (SBML) [14] that describes models of biological processes. Although the library is intended for biological processes, it has a complete translation tool for MathML that can be used in any domain.

In the semantic description, we have represented the interpolation prediction of the number of vehicles formula

[25] https://www.w3.org/TR/MathML3/.

[26] http://www.odaa.dk/dataset/realtids-trafikdata.

[27] http://iot.ee.surrey.ac.uk:8080.

as a MathML formula. Listing 5 shows an excerpt of the formula in turtle.[28]

```
form:formulaTrafficGrenavejarhusMorning rdf:type form:Formula ,
owl:NamedIndividual ;

form:hasFormulaValue """<?xml version=\"1.0\" encoding=\"UTF-8\"?>
<math xmlns=\"http://www.w3.org/1998/Math/MathML\">
 <apply>
  <plus/>
   <apply>
    <times/>
     <cn> 0.126603432701 </cn>
      <apply>
       <minus/>
        <ci> currentTimeInMinutes </ci>
        <cn type=\"integer\"> 180 </cn>
      </apply>
   </apply>
        <apply>
         <times/>
          <cn> 0.000765329644997 </cn>
          <ci> CurrentLocation </ci>
        </apply>
 </apply>
</math>""" .
```

Listing 5: Formula in the semantic description written in turtle.

Once we have the observations and formulas annotated in the semantic descriptions, when a user makes a query on the number of vehicles on the road between sensor 1 and sensor 3, both the value taken at sensor 1 and the formula is returned and then he can use the MathML expression to calculate estimated values for nearby locations or for future value predictions. In our example, we send a query from our application written in python to the triple store; read the last value taken at sensor 1 at 3:00 am GMT (which is around 5:00 am local time) together with the formula; convert the MathML into a python formula (using the library SBML in our python application); calculate the current value at our location (simulated to be at the place of sensor 2) and the current time (5:00 GMT); and get the expected number of cars at that position at that time, which is around 16 cars.

Dynamic semantics, therefore, can store spatio-temporal values in a triple store. Dynamic semantics has the advantage over other solutions (such as the use of a RESTful servers that calculate the current value from the formula) that all the information is stored in one place, in the triple store, giving a faster query-response time and a simpler service as the client only accesses one server. Our solution can also work in networks with low connectivity, as we can download the triple store when we have enough bandwidth and read it locally when needed.

8 Conclusions

In this study, we proposed a lightweight semantic IoT model, IoT-Lite. The model is an extension of SSN with shallow depth, appropriate for real-time sensor discovery. We have proposed and followed a set of ontology design guidelines for dynamic and responsive environments. We have demonstrated that the annotation of new sensors in IoT-Lite requires only 6 triples, and that the RTT of a query-response is in the range of milliseconds, even for large datasets. We have also assessed our proposal against another instantiation of SSN, IoT-A, and we have demonstrated that IoT-Lite performs better than IoT-A, in terms of memory requirements, computational time and RTT for a query-response, reducing the time by half for large datasets, such as for 100,000 sensors. We have also linked IoT-Lite with SAO ontology, which performs stream annotations allowing the aggregation of values, and therefore reducing the data values coming out from sensors. This solution can reduce the stream data triple store and reduce the query-response time for stream data. Furthermore, we have proposed dynamic semantic annotations to store formulas written in MathML into a triple store. We discussed an example of using dynamic semantics in a smart city and store spatio-temporal values in the triple store. This solution reduces the space used in the triple store and keeps all the information together in one place and therefore gives a faster query-response time.

Further work will provide IoT-Lite tools for annotation and validation, similar to SAOPY[29] and SSN validator [17]. We will also use the IoT-Lite based descriptions to provide interoperability in developing IoT and smart city applications and services. We will continue incorporating more functionalities to our dynamic semantics solution.

Acknowledgements The research leading to these results has received funding from the European Commission's in the Seventh Framework Programme for the FIWARE project under Grant Agreement No. 632893 and in the H2020 for FIESTA-IoT project under Grant Agreement No. CNECT-ICT-643943.

References

1. Baader F, Ghilardi S, Lutz C (2012) LTL over description logic axioms. ACM Trans Comput Logic (TOCL) 13(3):21
2. Barnaghi P, Bermudez-Edo M, Tönjes R (2015) Challenges for quality of data in smart cities. J Data Inf Qual (JDIQ) 6(2):6
3. Bermudez-Edo M, Noguera M, Hurtado-Torres N, Hurtado MV, Garrido JL (2013) Analyzing a firms international portfolio of technological knowledge: a declarative ontology-based owl approach for patent documents. Adv Eng Inform 27(3):358–365
4. Bermudez-Edo M, Elsaleh T, Barnaghi P, Taylor K (2015) Iot-lite ontology. W3C Member Submission. World Wide Web

[28] http://www.w3.org/TR/turtle/.

[29] http://iot.ee.surrey.ac.uk/citypulse/ontologies/sao/saopy.html.

Consortium (W3C), MIT Massachusetts, USA. http://www.w3.org/Submission/iot-lite/
5. Bonte P, Ongenae F, Schaballie J, De Meester B, Arndt D, Dereuddre W, Bhatti J, Verstichel S, Verborgh R, Van de Walle R, et al (2015) Evaluation and optimized usage of OWL 2 reasoners in an event-based ehealth context. In: 4e OWL reasoner evaluation (ORE) workshop. pp 1–7
6. Bornstein BJ, Keating SM, Jouraku A, Hucka M (2008) LibSBML: an API library for SBML. Bioinformatics 24(6):880–881
7. Botts M, Percivall G, Reed C, Davidson J (2008) OGC sensor web enablement: overview and high level architecture. In: GeoSensor networks. Springer, Heidelberg, pp 175–190. doi:10.1007/978-3-540-79996-2_10
8. Compton M, Barnaghi P, Bermudez L, García-Castro R, Corcho O, Cox S, Graybeal J, Hauswirth M, Henson C, Herzog A et al (2012) The SSN ontology of the W3C semantic sensor network incubator group. Web Semant Sci Serv Agents World Wide Web 17:25–32
9. Cox SJ (2016) Ontology for observations and sampling features, with alignments to existing models. Semant Web 8(3):453–470. doi:10.3233/SW-160214
10. De S, Barnaghi P, Bauer M, Meissner S (2011) Service modelling for the internet of things. In: Federated conference on computer science and information systems (FedCSIS). IEEE, pp 949–955
11. De S, Elsaleh T, Barnaghi P, Meissner S (2012) An internet of things platform for real-world and digital objects. Scalable Comput Pract Exp 13(1):45–58
12. Haller S (2010) The things in the internet of things. Poster at the (IoT 2010) Tokyo, November 5, p 26
13. Henson CA, Pschorr JK, Sheth AP, Thirunarayan K (2009) Semsos: semantic sensor observation service. In: International symposium on collaborative technologies and systems CTS'09. IEEE, pp 44–53
14. Hucka M, Bergmann F, Keating SM, Schaff JC, Smith LP (2010) The systems biology markup language (sbml): language specification for level 3 version. Nature proceedings
15. Janowicz K, Bröring A, Stasch C, Schade S, Everding T, Llaves A (2013) A RESTful proxy and data model for linked sensor data. Int J Digit Earth 6(3):233–254
16. Kolozali S, Bermudez-Edo M, Puschmann D, Ganz F, Barnaghi P (2014) A knowledge-based approach for real-time iot data stream annotation and processing. Internet of Things (iThings), IEEE International conference on, and green computing and communications (GreenCom). IEEE and cyber, physical and social computing (CPSCom) IEEE. IEEE, pp 215–222
17. Kolozali S, Elsaleh T, Barnaghi P (2014) A validation tool for the W3C SSN ontology based sensory semantic knowledge. Terra cognita and semantic sensor, networks. p 83
18. Kolozali S, Puschmann D, Bermudez-Edo M, Barnaghi P (2016) On the effect of adaptive and non-adaptive analysis of time-series sensory data. IEEE Internet Things J. doi:10.1109/JIOT.2016.2553080
19. López MF, Gómez-Pérez A, Sierra JP, Sierra AP (1999) Building a chemical ontology using methontology and the ontology design environment. IEEE Intell Syst 14(1):37–46
20. Müller H, Cabral L, Morshed A, Shu Y (2013) From RESTful to SPARQL: a case study on generating semantic sensor data. In: SSN@ ISWC. pp 51–66
21. Nati M, Gluhak A, Abangar H, Headley W (2013) Smartcampus: a user-centric testbed for internet of things experimentation. In: 16th International symposium on wireless personal multimedia communications (WPMC). IEEE, pp 1–6
22. Nati M, Gluhak A, Domaszewicz J, Lalis S, Moessner K (2014) Lessons from smartcampus: external experimenting with user-centric internet-of-things testbed. Wirel Personal Commun 1–15. doi:10.1007/s11277-014-2223-z
23. OneM2M (2014) Study of abstraction and semantics enablement v.0.7.0. study of existing abstraction and semantic capability enablement technologies for consideration by OneM2M. Technical Report OneM2M (TR 0007)
24. Pautasso C (2014) Restful web services: principles, patterns, emerging technologies. In: Web services foundations. Springer, New York, pp 31–51. doi:10.1007/978-1-4614-7518-7_2
25. Sanchez L, Muñoz L, Galache JA, Sotres P, Santana JR, Gutierrez V, Ramdhany R, Gluhak A, Krco S, Theodoridis E et al (2014) Smartsantander: IoT experimentation over a smart city testbed. Comput Netw 61:217–238
26. Sheth A, Henson C, Sahoo SS (2008) Semantic sensor web. IEEE Internet Comput 12(4):78–83
27. Stocker M, Shurpali N, Taylor K, Burba G, Rönkkö M, Kolehmainen M (2015) Emrooz: a scalable database for SSN observations. In: Joint proceedings of the 1st joint international workshop on semantic sensor networks and terra cognita (SSN-TC 2015) and the 4th international workshop on ordering and reasoning (OrdRing 2015) co-located with the 14th international semantic web conference (ISWC 2015), Bethlehem, 11–12 Oct 2015, pp 1–12. http://ceur-ws.org/Vol-1488/paper-01.pdf
28. Taylor K, Parsons E (2015) Where is everywhere: bringing location to the web. IEEE Internet Comput 19(2):83–87
29. Vermesan O, Friess P, Guillemin P, Gusmeroli S, Sundmaeker H, Bassi A, Jubert IS, Mazura M, Harrison M, Eisenhauer M et al (2011) Internet of things strategic research roadmap. Internet Things Glob Technol Soc Trends 1:9–25
30. Wang W, De S, Toenjes R, Reetz E, Moessner K (2012) A comprehensive ontology for knowledge representation in the internet of things. In: IEEE 11th international conference on trust, security and privacy in computing and communications (TrustCom). IEEE, pp 1793–1798