SPITFIRE: Towards a Semantic Web of Things

Dennis Pfisterer, Kay Römer, Daniel Bimschas, Henning Hasemann, Manfred Hauswirth, Marcel Karnstedt, Oliver Kleine, Alexander Kröller, Myriam Leggieri, Richard Mietz, Max Pagel, Alexandre Passant, Ray Richardson, and Cuong Truong

Abstract

The developed world is awash with sensors. However, they are typically locked into unimodal closed systems. To unleash their full potential, access to sensors should be opened such that their data and services can be integrated with data and services available in other information systems facilitating novel applications and services that are based on the state of the real world. We describe our vision and architecture of a Semantic Web of Things: a service infrastructure that makes the deployment and use of semantic applications involving Internet-connected sensors almost as easy as building, searching, and reading a Web page today.

The definitive version was published in the IEEE Communications Magazine 2011 (http://dx.doi.org/10.1109/MCOM.2011.6069708)

1 Introduction

Sensors are ubiquitous in infrastructures, appliances, mobile phones, and wireless sensor networks. Their widespread deployment represents a significant financial investment and technical achievement and the data they deliver is capable of supporting an almost unlimited set of high value proposition applications. This is a powerful and profitable confluence of need, capability, and economic opportunity – yet the true potential of sensor technology is massively under-exploited.

A central problem hampering success is that sensors are typically locked into unimodal closed systems. For example, motion detection sensors in a building may be exclusively controlled by the intrusion detection system. Yet the information they provide could be used by many other applications, e.g., placing empty buildings into an energy-conserving sleep mode or locating empty meeting rooms. Unlocking valuable sensor data from closed systems has the potential to revolutionize how we live. To realize this potential, a service infrastructure is needed to connect sensors to the Internet and publish their output in well-understood, machine-processible formats on the Web thus making them accessible and usable at large scale under controlled access.

So far, the sensor world and the Web world have been largely disconnected, requiring the human in the loop to find, integrate and use information and services from both worlds in a meaningful way. Publishing sensor-related data on the Web would help to find relevant information by directly accessing sensor data, i.e., by directly observing the real world, integrated with related information from the Web. Already today smart phone applications such as CenceMe exist that infer the activity of the person wearing the phone from sensor data and publish this in the Web. Another example are energy consumption sensors such as the Ploggs that end-users can install in their house to measure energy consumption of appliances, for example to compare their energy consumption with that of other, similar households to identify opportunities for saving energy. To do this easily, with open interfaces and data formats, and at large scale, technologies from the Web need to be customized for and integrated with their relevant counterparts on the Internet of Things (IoT). This means that application experts who are able to publish Web pages today should have the same easy-to-use technologies at hand to publish sensor descriptions, sensor data and make use of sensor outputs without requiring deep knowledge of embedded computing. In particular, we believe that users are primarily interested in real-world entities (things, places, and people) and their high-level states (empty, free, sitting, walking, ...) rather than in individual sensors and their raw output data. Therefore, the infrastructure has to provide appropriate abstractions to map sensors and their raw output to real-world entities with high-level states.

Real-world entities are rarely useful when considered in isolation – the ability to put multiple entities into a common semantic context is needed. For example, we want to reason about rooms being in the same building, belonging to the same company, with nearby parking spots. This requires a machine-readable representation of world knowledge and appropriate reasoning capabilities. Further, this representation needs to be unified - while most sensor data published so far on the Web relies on heterogeneous data models and serializations. In addition to discovery and query facilities on static properties of those machine-readable representations of sensors and real-world entities, specialized search approaches to support queries on the dynamically changing state of sensors or entities consisting of many sensors (possibly integrated with static data), will be required, e.g., which rooms in a building are currently occupied.

There are efforts to realize a Semantic Sensor Web including the SENSEI, SemSorGrid4Env, Exalted, and 52 North projects, as well as work by the Kno.e.sis Center, CSIRO, and the Spanish Meteorological Agency. Most notably, the Open Geospatial Consortium's (OGC) Sensor Web Enablement (SWE) [1] project builds a framework to publish and access sensor data using XML-based protocols and APIs. The choice of XML, however, ties SWE to system-specific schemas, providing neither semantic interoperability nor a basis for reasoning. This problem is in the focus of the Semantic Sensor Web [2] which proposes annotating sensor data with semantic meta-data, whose meaning is machine-understandable through vocabulary definitions, i.e., an ontology. By annotating sensor-related features such as the network, deployment, data formats, etc., it becomes possible to automate further tasks, e.g., deployment, maintenance, and integration.

However, these efforts have limitations which we are addressing in SPITFIRE: There is no general-purpose approach compatible with the growing body of semantic world knowledge available as Linked Open Data (LOD) on the Web; existing efforts are either too sensor-centric or too knowledge-centric, i.e., they do not provide comprehensive, integrated abstractions for things, their high-level states, and how they are linked to sensors; and a number of important services are missing in existing efforts, notably support for semi-automatically creating Linked Data representations of sensors and things, as well as efficient search for things based on their current states.

SPITFIRE addresses these limitations by providing i) vocabularies to integrate descriptions of sensors and things with the LOD cloud, ii) semantic entities as an abstraction for things with high-level states inferred from embedded sensors, iii) semi-automatic generation of semantic sensor descriptions, and iv) efficient search for sensors and things based on their current states. In addition, SPITFIRE integrates these ingredients into a unified service infrastructure to ease adoption of the Semantic Web of Things for end-users and developers. On top of this infrastructure, applications are assembled by issuing search requests for matching (real or aggregated) sensor services and by invoking found services directly.

This paper proceeds with the description of an exemplary use case that will be used to illustrate the state of the art with respect to integration of sensors into the Web upon which SPITFIRE builds, followed by a description of the novel contributions of SPITFIRE and a brief discussion of an existing operational prototype.

2 Use Case and Requirements

Due to an emergency, a traveling salesman drives to his company headquarters to hold an ad-hoc meeting. For that, he must find a currently free room in the headquarters that are dispersed over a large area in order to hold an ad hoc meeting. After that, he informs his colleagues and searches for a parking spot close to the building.

Imagine that sensors, which are connected to the Internet, measure the state of real-world entities such as meeting rooms and parking spots. Internet-connectivity not only requires network-level integration (IP), but also application-level integration to enable structured access to sensor data. To enable automatic reasoning about sensors (e.g., finding free parking spots close to meeting room), these sensors, their output, and their embedding into the real world must be *described in a machine-readable format* that is compatible with data formats used to describe existing world knowledge in the Web. Not only syntax and semantics of such a description must be defined, but efficient mechanisms to *annotate newly deployed sensors* with appropriate descriptions are required.

Users are primarily interested in *real-world entities* (e.g., meeting room) and their *high-level states* (e.g., room occupied) rather than sensors (e.g., sensor 536) and their raw output (e.g., motion detected at time T). Therefore, appropriate mechanisms to establish an explicit mapping of sets of sensors to real-world entities they are monitoring (e.g., all motion detection sensors in a certain room) must be provided. Further, the raw output of these sensors (e.g., motion detection events) has to be mapped to a high-level state (e.g., room occupied). Often, this involves fusing the output of multiple sensors (e.g., multiple motion sensors are needed to cover a large room) or even scheduling sensors for energy efficiency (e.g., only one out of two available battery-powered motion sensors is required to cover a smaller room).

Finally, the user wants to *search for real-world entities by their current state* (e.g., empty meeting rooms). Often, such search requests refer not only to the output of sensors, but also to further machine-readable information that is available elsewhere in the Web (e.g., company maps, meeting schedules, calendars). The search engine needs to integrate these different static and dynamic data sources in a seamless way.

3 Semantic Sensor Web: State of the Art

Realizing the above use case on an Internet scale requires i) that the sensors are connected to the Internet, ii) that machines can discover and understand the semantics of the data returned by the sensors, and iii) a technique to find the sensors that could provide the relevant data. This section briefly discusses the state-of-the-art in relation to this with a focus on Internet-scale, Web-based technologies upon which SPITFIRE builds, employing the use case as an example.

3.1 Connecting Sensors to the Internet and the Web

Integrating resource-constrained sensors into the Internet is difficult since ubiquitously deployed Internet protocols such as HTTP, TCP or even IP are too complex and resource-demanding. To achieve integration, light-weight alternatives are required that can easily be converted from/to Internet protocols.

Only recently, two such alternatives are gaining momentum: 6LoWPAN and CoAP. 6LoWPAN [3] is a light-weight IPv6 adaptation layer allowing sensors to exchange IPv6 packets with the Internet. Currently, only UDP is specified as TCP is considered too resource consuming. CoAP (Constrained Application Protocol [4]) is a draft by IETF's CoRE working group, which deals with Constrained RESTful Environments. It provides a light-weight alternative to HTTP using a binary representation and a subset of HTTP's methods (GET, PUT, POST, and DELETE). In addition, CoAP provides some transport reliability using acknowledgements and retransmissions. For a seamless integration, reverse proxies may convert 6LoWPAN and CoAP to TCP and HTTP so that sensor data can be accessed using these omnipresent protocols. Also, Internet-based clients could directly use CoAP on top of UDP.

6LoWPAN in combination with CoAP allows sensors to be queried from the Internet as they can provide so-called RESTful web services. Those are services following the Web's REST (REpresentational State Transfer) principles [5]. Resources (e.g., sensors) are addressed using standard URIs and data can be returned in different representations (e.g., HTML or RDF) using HTTP content negotiation.

RESTful services are queried and manipulated using the aforementioned four HTTP methods. For instance, an application could query the state of a sensor by sending a GET request to the sensor (e.g., http://ipv6-address-or-dns-name/room-sensor. The sensor replies with its value encoded in a, possibly proprietary, encoding (e.g., in plain text: "*occupied*" or any format the sensor supports). For an exhaustive discussion of 6LoWPAN, CoAP, and RESTful services, we refer the reader to [6].

To discover the services hosted on a CoAP server, the CoRE Link Format specification defines how Web Linking described in RFC5988 is used by CoAP servers. Clients use a well-known URI (/.well-known/core) to retrieve a list of resources. For instance, the room sensor device could return *</room-sensor>;ct=0;rt="ex:RoomSensor"* to indicate that the resource /room-sensor returns the content type *text/plain* (indicated by $ct = 0$) and that the resource type is *ex:RoomSensor*. The latter is a concept from an ontology (e.g., the W3C SSN-XG sensor ontology to which some of the authors contributed) as described in the following section. Note that concepts appearing in different ontologies can be automatically mapped[1].

RESTful services (i.e., the operations provided, their parameters and return values) can be described using, for example, Web Service Definition Language (WSDL) version 2.0. For example, RESTful versions of OGC's Sensor Observation Services have been proposed and are currently under consideration by the Sensor Web Enablement group.

3.2 Linked Sensor Data

The integration of sensors into the Internet using CoAP/HTTP already enables many applications in which developers query and process data provided by a well-known set of sensors. However, such manual integration does not scale. What is required is a "machine-understandable" description of sensors and the data they produce. Semantic Web [7] technologies fulfill this requirement as they enable machines to understand, process, and interlink data using structured descriptions of resource and Linked Open Data as the framework makes this integration both immediate and meaningful through the inclusion of semantic links into a resource's machine-readable description.

The predominant technique for *machine-readable representations of knowledge* on the Web is the Resource Description Framework (RDF), which represents knowledge as *(subject, predicate, object)*-triples (e.g., *Sensor3 is-in ParkingSpot41* or *ParkingSpot41 is-in Berlin*). A set of triples forms a graph where subjects and objects are vertices and predicates are edges. From the graph formed by these two triples, one can infer that *Sensor3* is in Berlin by exploiting the knowledge (contained in so-called ontologies) that *is-in* is a transitive property. Such knowledge is often expressed using OWL (Web Ontology Language), one of the main languages (with RDF Schema) to define ontologies on the Web.

1. http://www.w3.org/DesignIssues/RDF-XML

It is imperative to use non-ambiguous identifiers for subjects, predicates, and objects to guarantee uniqueness on an Internet-scale, which is achieved by encoding them as URIs. The above triple could be expressed as follows:

- a subject (<http://example.com/sensors/sensor3>),
- a predicate (<http://www.loa-cnr.it/ontologies/DUL.owl#hasLocation>), and
- an object (<http://example.com/parkingSpot/spot41>).

The Linked Data model does not enforce special URIs but encourages the use of widely-used URIs so that a densely interlinked graph emerges. *Ontologies* play an important role in defining the URIs for a specific application domain and their relation to each other as they "standardize" agreed, conceptual knowledge. For example, an ontology could define a generic *sensor* (e.g., http://purl.oclc.org/NET/ssnx/ssn#Sensor), an occupancy detection sensor (e.g., http://example.com/ontology/spitfire.owl#Occupancy), and define that occupancy sensor is a sub-class of sensor, which creates a relation between the two URIs. Semantic search engines queried for *sensors* at a certain location could therefore specifically return information on *occupancy detection sensors*. The CoAP link-format we use (RFC5988) allows to specify URIs eventually pointing to semantic definitions, i.e., support for semantic annotation of links inside a sensor network. Additionally, to make these semantic descriptions available on the Web, we could imagine to annotate pages describing sensors using RDFa or SA-REST, so that the same document is used for humans and machines.

3.3 Search for Sensors

Assuming that sensors are described by such RDF triples, a search service can find sensors based on meta-data such as sensor type, location, or accuracy. For instance, applications could ask for parking spots in Berlin to calculate the city's availability of car parking places. Such queries can be expressed in SPARQL and the aforementioned question could be answered using the (simplified) SPARQL query in Figure 1. In the query, question marks indicate variables (e.g., "node" and "spot"), while "spots" is an aggregate value.

```
SELECT COUNT(DISTINCT ?node) as ?spots
WHERE {
 ?node a ssn:Sensor ;
   ssn:observes ex:Occupancy ;
   dul:hasLocation ?spot .
 ?spot a ex:ParkingSpot ;
   dul:hasLocation dbpedia:Berlin .
}
```

Fig. 1. SPARQL query requesting all occupancy sensors located at parking spots in Berlin.

The variables in a SPARQL query are matched against triples in databases (triple stores) and are bound to the matching fields in the matching triples. That is, the query finds subjects that are sensors observing occupancy that are located in a spot that is a parking spot located in Berlin. There are a number of existing efforts to support semantic sensor discovery but they are not as comprehensive as in SPITFIRE. E.g., [8] does not expose Linked Data and [9] does not exploit the hierarchical and structured relations which are relevant even for such simple queries as above. To further exploit these annotations, we could also use faceted browsers such as MIT Simile's Exhibit, where facets for identifying parking places could be location, availability, but also static information such as price-range

4 From Semantic Sensor Web to Semantic Web of Things

The techniques for integration of sensors into the Web outlined in the previous section are necessary, but not sufficient to realize a Semantic Web of Things. In particular, semantic descriptions must be integrated with the LOD cloud to support semantic reasoning; semantic descriptions need to be semi-automatically created for sensors and things to allow end-users to use this technology at scale; abstractions for things, their high-level states, and integration with sensors are required; and search for things with a given current state is needed. In this section, we show how SPITFIRE addresses these requirements.

4.1 Advanced Semantics of Sensors

Our main contribution in semantics for sensors over projects such as SENSEI and SemSorGrid4Env are:

- scalability through avoiding any registry of semantic entities;
- linkage with the LOD cloud is not restricted to a predefined set of datasets (as in Kno.e.sis' Real Time Feature Streams and DERI's SensorMasher);
- generation of semantic annotations from raw data (as in SENSEI) but improved by searching for already existing concepts to reuse and datasets to link to.

To facilitate this, we developed an ontology starting from the alignment (i.e., mapping equivalent concepts in different ontologies such as *room* and *chamber*) of the already existing ontologies such as *Dolce Ultralite*, the W3C Semantic Sensor Network (SSN-XG) ontology (based on OGC's Observations & Measurements and Sensor Model Language), and the *Event Model F*, to support cross-domain descriptions of sensor-related data and its context (higher-level events).

As an example, Figure 2 presents part of an RDF description of an occupancy detection sensor located in a parking spot. The triples state that a particular object is a sensor measuring occupancy located in a particular parking spot. The parking spot belongs to a specific company and is located in a given area with a given geographical location (triples following a semicolon have only a predicate and object – the subject is the same as in the previous triple). This shows how a sensor could provide an unambiguous machine-understandable self-description.

```
<http://sensorportal.tu-braunschweig.de/ontology/spitfire.owl#sensor1>
 rdf:type ssn:Sensor ;
 ssn:observes ex:Occupancy ;
 dul:hasLocation <http://sensorportal.tu-braunschweig.de/parkingSpot/ACMEParkingPlace1> .

<http://sensorportal.tu-braunschweig.de/parkingSpot/ACMEParkingPlace1>
        rdf:type ex:parkingSpot ;
 dul:isPartOf <http://sensorportal.tu-braunschweig.de/building/plan/companyPlan1> ;
 geo:lat "51" ;
 geo:long "0.4" ;
 dul:hasLocation <http://sensorportal.tu-braunschweig.de/parkingArea/area10> ;
 ssn:attachedSystem <http://sensorportal.tu-braunschweig.de/ontology/spitfire.owl#sensor1> .
```

Fig. 2. Exemplary description of a parking spot occupancy detection sensor.

SPARQL can be used for search on top of such descriptions but cannot be used for sensor output directly as this would require that new triples are stored and indexed by a triple store whenever the output of a sensor changes. However, SPARQL compatible support for this is under active development in SPITFIRE (Continuous Query Evaluation over Linked Streams – CQELS).

A very important feature of our approach is that arbitrary datasets on the LOD can be dynamically linked in order to describe complex real-world processes and to detect facts and correlations about them. For instance, the above parking spot sensors indirectly also provide information about environmental pollution caused by cars, which can be integrated with additional environmental features from http://linkedgeodata.org/, and abnormally high death rates of local population from http://www4.wiwiss.fu-berlin.de/eurostat/. All publications about such features can be selected from http://eprints.rkbexplorer.com/. Specific information for certain countries such as the UK is also available on the LOD from http://education.data.gov.uk.

4.2 Semi-Automatic Creation of Semantic Sensor Descriptions

Besides a language for semantically describing sensors and their output, as well as their embedding into the real world, a mechanism to annotate newly deployed sensors with descriptions in this language is required. The technically simplest way is to let humans provide these but this approach does not scale as end-users typically cannot be expected to provide such semantic descriptions of sensors and their deployment contexts. Therefore, we developed a semi-automatic approach for annotating newly deployed sensors under the hypothesis that sensors with similar semantic descriptions would produce similar output. For example, two motion sensors deployed in the same room should produce similar time series. Assume that one sensor is manually annotated with its room number and later, a second sensor is deployed in the same room. If the output of this sensor correlates over some time with the first sensor, we can conclude that they are in the same location and copy the annotation. Based on the strength of the correlation we can compute the confidence of the correctness of the inferred annotation. If there is no sensor with a strong correlation, then the user must provide the annotation manually. If there is one sensor with a strong correlation, the user must confirm the use of this annotation. If there are multiple sensors with different annotations but similar correlation strength, then user must chose among those (assuming that users have access to a computer with appropriate user interfaces). That is, initially users must provide the annotations manually to bootstrap the system. Over time, more and more sensors are annotated, increasing the chances of our algorithm correctly annotating new sensors. Specifically, if multiple sensors with the same annotation but slightly different output time series exist, a voting mechanism may improve the confidence.

The underlying scientific problem is the classification of sensor data streams, which can be solved by means of data clustering as illustrated in Figure 3. Sensors with the same known annotation form a cluster. When a new sensor with unknown annotation appears, we compute a distance measure to all known clusters, assume the new sensor belongs to the closest cluster and copy the cluster annotation to the new sensor. However, if the distance to the closest cluster is too large or if there is a tie between multiple clusters, the user is involved as described above. The distance between two sensors is computed by filtering and normalizing the raw sensor output over a given time window, and by computing a distance metric between pairs of sensors. Currently, we use a fuzzy approach for the distance computation, where an output time series of a reference sensor is mapped to fuzzy sets, and the output time series of a newly deployed sensor is matched against those fuzzy sets to compute a similarity score. This algorithm does not only take into account the possible sensor output values, but their distribution over time. This approach can not only successfully classify sensors according to their type (i.e., measured physical quantity), but also according to different measurement units for the same physical quantity (e.g., degrees Celsius vs. degrees Fahrenheit), and according to location (e.g., room where sensors are located).

4.3 Semantic Entities: An Abstraction for Things

As motivated in Section 2, users are interested in real-world entities and their high-level states rather than in raw sensor output. We propose semantic entities as a concept to map sensors and their raw output to real-world entities with high-level states. That is, semantic entities provide

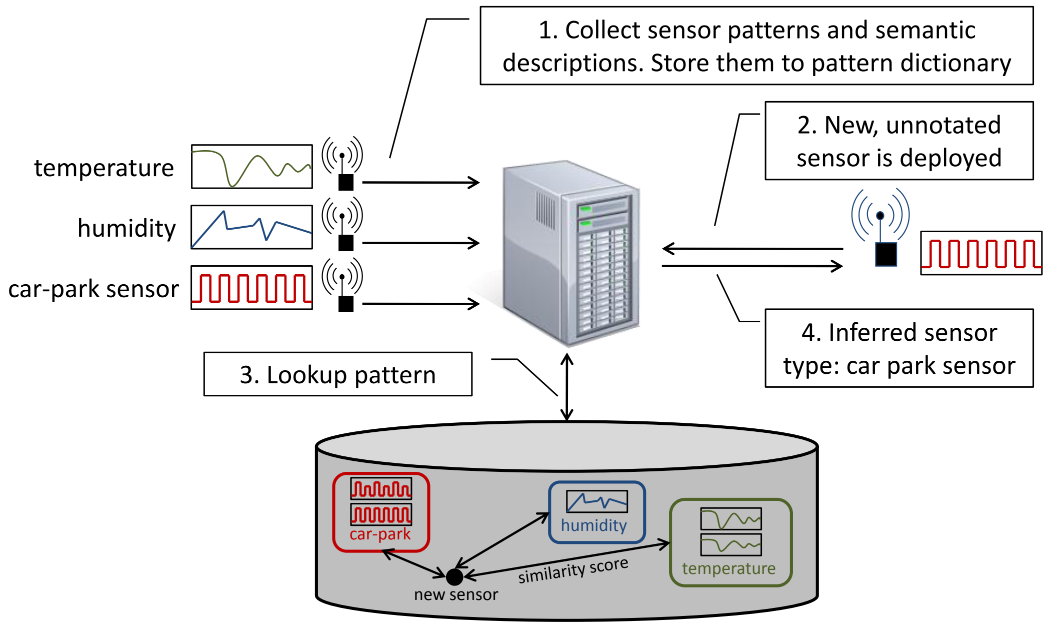
1. Collect sensor patterns and semantic descriptions. Store them to pattern dictionary
2. New, unnotated sensor is deployed
temperature
humidity
car-park sensor
3. Lookup pattern
4. Inferred sensor type: car park sensor
car-park
humidity
temperature
similarity score
new sensor

Fig. 3. Clusters of annotated sensors and automatic inference of the annotation of a new sensor.

two orthogonal functions: mapping sensors to real-world entities and computing high-level states from raw output of one or more of those sensors.

This mapping is accomplished using SPARQL queries. For example, to construct a semantic entity for a parking spot, one would query for sensors (e.g., magnetometers to detect the presence of metal) located at this parking spot. Additionally, the output of these sensors has to be mapped to the high-level states "occupied" or "free." For that, we assume the existence of a RESTful Web Service that computes the high-level state from raw sensor output. We call such a Web Service a *virtual sensor*. Using a PUT operation, the virtual sensor can be configured to accept input from the previously found real sensors and the computed high-level state can be retrieved by a GET operation from the virtual sensor in the same way as retrieving the output of a real sensor.

The newly created semantic entity is represented by a set of RDF triples describing its type and location, as well a providing pointers to the virtual sensor(s) computing its high-level state(s). Those virtual sensors in turn are also described by RDF triples so that they can be found by SPARQL queries.

Initially, virtual sensors have to be implemented manually, but over time a large set of reusable virtual sensors for common high-level states will be created and can be reused by posing SPARQL queries to find them. The virtual sensors either live on physical sensor nodes (avoiding the transmission of raw sensor time series across the Internet) or on Internet servers (if state computation exceeds the resources of a sensor node). If the real-world entity being represented by the semantic entity is covered by multiple sensors in a redundant fashion, the virtual sensor may also disable redundant sensors to save energy resources.

Figure 4 (i) shows a SPARQL query to find all sensors located at parking spot "ACMEParkingPlace1". Subfigure (ii) shows the resulting set of RDF triples describing the semantic entity created for this parking spot. The reference to the newly created virtual occupancy sensor whose RDF description is shown further below, containing a link back to the semantic entity.

4.4 Searching Things by Their Current States

An important functionality required by the use case sketched in Section 2 is searching for semantic entities that exhibit a certain high-level state at the time of the query. As the high-

(i)

```
SELECT DISTINCT ?node
WHERE {
 ?node a ssn:Sensor ;
   dul:hasLocation ex:ACMEParkingPlace1 .
}
```

(ii)

```
<http://example.org/se/0d59-9f0e-c05c-3bda>
 rdf:type ex:ParkingSpot ;
 dul:hasLocation ex:ACMEParkingPlace1 ;
 ssn:attachedSystem <http://sensorportal.tu-braunschweig.de/ontology/spitfire.owl#sensor1> .

<http://sensorportal.tu-braunschweig.de/ontology/spitfire.owl#sensor1>
 rdf:type ssn:Sensor ;
 ssn:observes se:Occupancy ;
 dul:isPartOf <http://example.org/se/0d59-9f0e-c05c-3bda> .
```

Fig. 4. (i) Example SPARQL query to find sensors at a given parking spot. (ii) Example set of RDF triples describing the semantic entity and its virtual occupancy sensor. The ParkingSpot class is a sub-class of se:SemanticEntity.

level states typically change very frequently, SPARQL is not directly applicable. Even CoAP's observe function to report changes of a resource rather then streaming sensor values would not help as all sensors in the world would then push updates at high frequencies, which does not scale to an Internet of Things. Even installing an observation filter such that only values sought by a query are reported does not help, as then all relevant sensors would have to be tasked for each one-short query. We address this challenge by developing heuristics to efficiently identify entities that are *likely* to match a given search. Only for those likely candidates, we retrieve their actual current state to check if they match [10].

We employ prediction models to compute the probability that the current high-level state of a semantic entity matches the value we are searching for (e.g., parking spots that are *empty*). These prediction models are created by the virtual sensors (this is especially efficient if the virtual sensor lives on the physical sensor node that generates the input sensor data for the virtual sensor) and are periodically indexed by a search engine. When a search is performed, the search engine executes the indexed prediction models to obtain the matching probability without communicating with the virtual sensor. The semantic entities are then sorted by decreasing probability and contacted in this order to fetch their actual current state until enough matching entities have been found. We integrate this approach into SPARQL by encoding the prediction models as RDF triples and evaluating them in the SPARQL query. Using the ORDER BY clause, entities are then sorted by decreasing probability.

We employ different types of prediction models. The first type of model exploits periodic patterns in past states (e.g., a meeting room is occupied every Monday from 8 to 10). A second type of model exploits correlations of sensors (e.g., parking spots close to the entrance of a building are often all occupied, whereas spots further away are often free), so by knowing the occupancy of a parking spot one can infer the state of a neighboring parking spot with high probability. The prediction models we consider are based on simple statistics and computational overhead for creating and executing them is small.

Figure 5 (i) shows an extended version of the occupancy virtual sensor entity of Figure 4 (ii), encoding a very simple prediction model (i.e., the spot was occupied with probability 0.2 and free with probability 0.8 in the past). The SPARQL query in Subfigure (ii) returns a list of parking-spot semantic entities sorted by decreasing probability of being free according to the above prediction model.

(i)

```
<http://sensorportal.tu-braunschweig.de/ontology/spitfire.owl#sensor1>
 rdf:type ssn:Sensor ;
 ssn:observes se:Occupancy
 dul:isPartOf <http://example.org/se/0d59-9f0e-c05c-3bda> ;
 ex:FreeStateHasProbability "0.8" ;
 ex:OccupiedStateHasProbability "0.2" .
```

(ii)

```
SELECT DISTINCT ?entity
WHERE {
   ?entity a ex:ParkingSpot ;
       ssn:attachedSystem ?node .
   ?node a ssn:Sensor ;
       ex:FreeStateHasProbability ?prob .
} ORDER BY ?prob
```

Fig. 5. (i) RDF encoding of a simple prediction model for a virtual sensor. (ii) SPARQL query to compute a ranked list of parking-spot semantic entities.

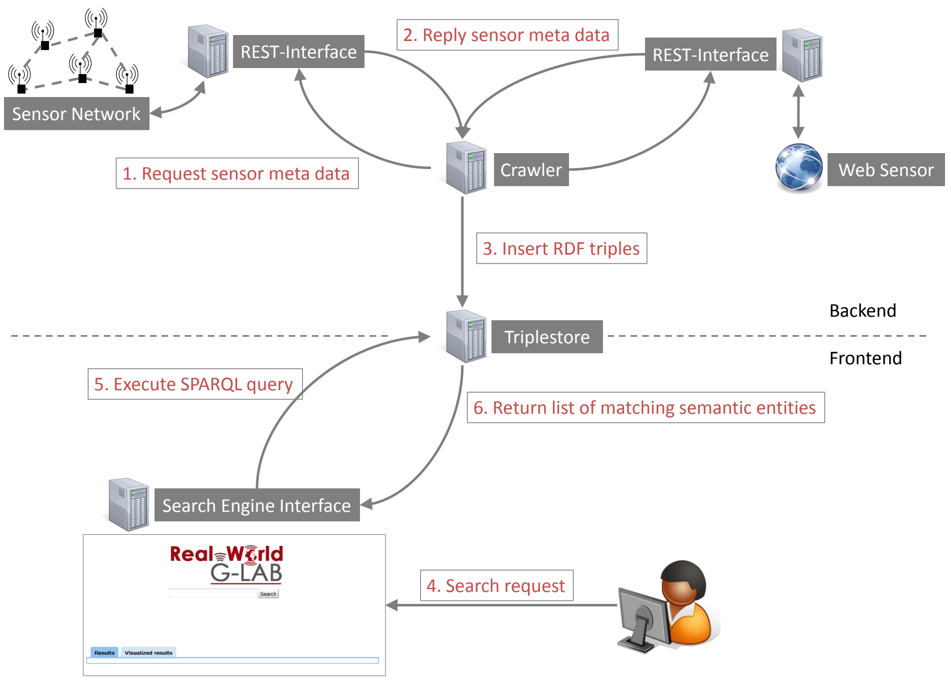

Fig. 6. Architecture of the reference implementation.

5 Reference Implementation

To demonstrate the feasibility of the proposed architecture, we realized a prototype using the Jena Semantic Web Framework. Figure 6 shows an overview of the architecture along with the steps involved in performing a search operation.

An RDF triple store (i.e., a distributed or grid/cloud-based database storing RDF triples) forms the core of the system. A crawler periodically scans the Semantic Web of Things for semantic entities and sensors, downloads meta-data and prediction models using their RESTful web interfaces, converts this information into RDF triples (if necessary) and stores them in the triple store. iSense sensor nodes equipped with motion and temperature sensors are deployed in our office building experiment. Their raw output is converted into a high-level state reflecting room occupancy by a virtual sensor. The sensor nodes run 6LoWPAN/IPv6 on the network layer and an implementation of the current IETF CoAP Draft on the transport layer to provide a RESTful interface to the outside world. A proxy translates between CoAP/UDP-based service invocations and RESTful HTTP Web Service calls that are being issued by the crawler or arbitrary client applications. In addition to physical sensors, our system supports so-called *web sensors*, i.e., sensors whose output is published on Web pages that are parsed by the crawler to extract sensor values. In particular, we are using a city-wide online parking web application to obtain real-time information about parking lot occupancy throughout the city.

Semantic data stored in the triple store can be queried using a SPARQL query engine. Our aim is to also support non-expert users who would be unable to formulate SPARQL queries. We therefore offer a simple graphical user front end which accepts a keyword-based query and converts it into a SPARQL query at the cost of reduced query expressiveness. A map-based interface shows locations of the matching semantic entities.

The system prototype is operational and implements the use case described in Section 2. The system accesses about 40 physical sensors (temperature, movement detectors) attached to 20 sensors nodes in our office building, as well as 25 web sensors (parking lot occupancy). We have also studied the efficiency of the underlying content-based search algorithm in [10], showing that the communication overhead is reduced by up to one order of magnitude compared to a pull-based baseline.

Future steps include (distributed / grid / cloud-based) triplestore scalability for sensor data, which uses domain knowledge about the stored triples and specialized data placement policies to optimally distribute triples. Also, we are working on supporting better reasoning by making use of LOD ontologies in queries. Finally, we are currently working on integrating a service for semi-automatic sensor annotation into the system. The reference implementation will be provided at http://spitfire-project.eu.

6 Conclusions

Existing Semantic *Sensor Web* technologies enable the integration of sensors into the Web, but the underlying model is focused on sensors rather than on things and their high-level states. Existing approaches also lack integration with the LOD cloud, a quickly growing and open base of semantic world knowledge. With SPITFIRE, we work towards a Semantic *Web of Things*, by providing abstractions for things, fundamental services for search and annotation, as well as by integrating sensors and things into the LOD cloud. Using Linked Data principles makes sensor data easily accessible for applications via existing mechanisms deployed on the Web, which will significantly speed up the uptake of IoT technologies. We demonstrated the feasibility by an operational prototype that was used to realize a representative use case. As SPITFIRE is in line with the most promising developments on both the IoT and Web sides, the system will benefit from any advances made in these areas implicitly.

As it was difficult to foresee the wealth of current Web applications back when the Web was created, we have to wait and see how people will use the Semantic Web of Things. It is also hard to predict if a Semantic Web of Things will be as broadly adopted as the Web is today. One indicator is that LOD has already achieved significant uptake by governments (including UK, USA), the media sector (BBC), life sciences, geo information systems, and Web companies (Freebase). Making sensor data part of this data pool is clearly beneficial as then integration with knowledge from arbitrary sources is possible. For example, sensors and their data can be linked to geographic data (correlated natural phenomena), user-generated data (social feedback), government data (census information), life-science data (causes and effects of diseases), etc.

A strong indicator whether this line of development will be successful in the long run, is also provided by the exponentially growing amount of linked data and the support by major players. Since its beginnings in 2007, the LOD cloud has grown from 12 datasets to 203 data sets in 2010 with over 25 billion triples interlinked with 395 million links. Industry initiatives such as Google's Rich Snippet (2009), Facebook's Open Graph (2010), or very recently Schema.org (2011) all aim at adding semantic markups to web pages to improve search and discovery capabilities for the end user also confirm this uptake at Web-scale.

References

[1] OGC - Open Geospatial Consortium, "Sensor Web Enablement (SWE)," 2010.

[2] A. Sheth, C. Henson, and S. Sahoo, "Semantic Sensor Web," *IEEE Internet Computing, July/August 2008*, 2008, pp. 78–83.

[3] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," RFC 4944 (Proposed Standard), Sep. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4944.txt

[4] Z. Shelby, K. Hartke, C. Bormann, and B. Frank, "Constrained application protocol (CoAP) (CoRE working group)," Online version at http://www.ietf.org/id/draft-ietf-core-coap-06.txt (07/07/2011), 2011.

[5] R. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, Irvine, 2000.

[6] Z. Shelby, "Embedded web services," *Wireless Communications, IEEE*, vol. 17, no. 6, Dec. 2010, pp. 52–57. [Online]. Available: http://dx.doi.org/10.1109/MWC.2010.5675778

[7] T. Lee, J. Hendler, O. Lassila *et al.*, "The semantic web," *Scientific American*, vol. 284, no. 5, 2001, pp. 34–43.

[8] S. Jirka, A. Broring, and C. Stasch, "Discovery Mechanisms for Sensor Web," in *Sensors*, 2009.

[9] J. Pschorr, C. Henson, H. Patni, and A. P. Sheth, "Sensor discovery on linked data," Kno.e.sis, Tech. Rep., 2010.

[10] B. M. Elahi, K. Römer, B. Ostermaier, M. Fahrmair, and W. Kellerer, "Sensor ranking: A primitive for efficient content-based sensor search," in *Proceedings of the 2009 Intl. Conference on Information Processing in Sensor Networks*, 2009, pp. 217–228.

Acknowledgments

This work has been partially supported by the European Union under contract number ICT-2009-258885 (SPITFIRE), by Science Foundation Ireland under Grant No. SFI/09/CE/I1380 (Líon2), and by the Federal Ministry of Education and Research of Germany (Förderkennzeichen 01BK0905, Real-World G-Lab).