IoT Interoperability: A Hub-based Approach

Mike Blackstock
University of British Columbia
Vancouver, Canada
mblackst@magic.ubc.,ca

Rodger Lea
University of British Columbia, Vancouver, Canada
& University of Lancaster, UK
rodgerl@ece.ubc.ca

***Abstract*—Interoperability in the Internet of Things is critical for emerging services and applications. In this paper we advocate the use of IoT 'hubs' to aggregate things using web protocols, and suggest a staged approach to interoperability. In the context of a UK government funded project involving 8 IoT projects to address cross-domain IoT interoperability, we introduce the HyperCat IoT catalogue specification. We then describe the tools and techniques we developed to adapt an existing data portal and IoT platform to this specification, and provide an IoT hub focused on the highways industry called 'Smart Streets'. Based on our experience developing this large scale IoT hub, we outline lessons learned which we hope will contribute to ongoing efforts to create an interoperable global IoT ecosystem.**

Keywords—internet of things; interoperability

I. INTRODUCTION

The Internet of Things (IoT) in which everyday objects can be equipped with identifying, sensing and processing capabilities, and then connected to the Internet, promises significant benefits both in terms of efficiencies and new services [1]. To reach the full potential of the IoT, however, it is not sufficient for things to just be connected to the Internet; they also need to be found, accessed, managed and potentially connected to other 'things'. To enable this interaction, a degree of interoperability is necessary that goes beyond simple protocol interoperability as provided by the Internet.

As we strive for greater interoperability, one logical next step is to exploit web technologies such as HTTP, JSON and the Representational State Transfer (REST) architecture of the World Wide Web, an approach referred to as the 'Web of Things' [2]. This use of the web provides a higher degree of interoperability, potentially connecting islands of things in different domains. By doing so, developers can connect things using web tools and technologies and create new applications and mashups [3], [4] that combine data from physical things in different domains with other online services such as social networks [5], [6].

However, even as many in the IoT community have converged on the use of web technologies, the plethora of systems for the Web Of Things testifies to the fact that there is no standard approach to exposing physical objects to the web today. To address this issue, some have begun to create large-scale 'hubs' that provide a consistent and easy-to-use interface for both integrators and application developers (e.g. [3], [7], [8]). These hubs provide facilities for search, for (meta) data storage and for interaction between things and applications.

While these hubs, by the simple process of aggregation of data and standard representation of 'things' provide a degree of interoperability, they typically do not inter-operate with each other [9]. We argue that this lack of inter-hub interoperability may stifle the uptake of the IoT. Many of the things of interest to application and service developers will only be accessible using product or hub-specific APIs. Long term, this must be addressed through a standardization process, and indeed many in the web and IoT community have started this process in various groups (e.g. [10], [11], [12], [13]). While this is important work that must continue, we argue that it will be difficult to achieve consensus until the fundamental requirements for these IoT hubs are clearly established. Premature standardization could risk stifling innovation. At the same time, and even as the community evolves, there is a need for some degree of interoperability, if we are to offer developers more than simple islands of Internet connected-things, requiring developers to address interoperability issues themselves.

II. PATH TO HUB INTEROPERABILITY

In previous work [9] we proposed a four stage path toward greater interoperability between IoT hubs.

1. IoT Core. Hubs expose things and associated metadata using the web architecture and RESTful web services _ a web of things.

2. IoT Model. Agreement on basic approaches and models requiring a common understanding of what things and associated data a hub should contain. Achieving this stage will facilitate the development of adapters and other integration tools for hub interoperability.

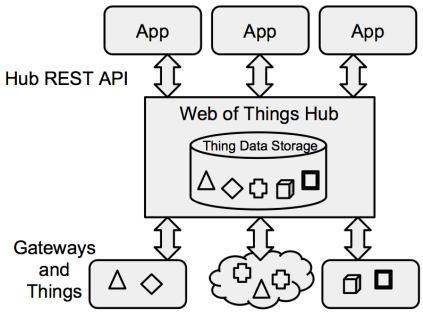

Fig. 1. IoT hubs provide single consistent interface for hosted 'things'

3. IoT Hub. Agreement on certain implementation issues such as concrete representations, URLs and schema for describing and querying catalogs and data from hubs. This will include support for security mechanisms so that hubs can control access to things and offer some guarantees over who is providing things and their data and who is able to access and use these resources.

4. IoT Profiles. Agreement on the semantics of things and their associated data exposed on a hub. For example: a temperature sensor in one hub provides the same quality and value of temperature as one in another hub. Essentially the taxonomy of things and the ontological models that hubs support will need to be defined. By reaching agreement at this level, deep integration of application is possible, allowing hubs and things to link to and communicate directly with each other.

Assuming that hubs are exposing IoT resources using web protocols (stage 1), we need to begin moving toward agreement on basic models for the IoT (stage 2) and agreement on hub implementation issues (stage3). To that end, we have begun to explore the use of integration tools to map existing IoT systems to emerging specifications and diverse data sources. This work was performed in the context of a UK funded IoT programme that established 8 IoT data hubs from different domains and explored the creation of an interoperability specification called HyperCat [14].

In this paper, we briefly describe the HyperCat specification that aims to unify the catalogues of these hubs with a shared representation and query mechanism. This is followed by a description of the tools we have developed to ease the implementation tasks we faced when we attempted to expose our own IoT catalogues as specified by HyperCat and tried to integrate disparate data sources into our IoT hub for access by developers in a uniform manner. We conclude with some lessons from our experiences building a large-scale IoT hub and supporting the HyperCat approach to interoperability, hoping that our experiences will contribute to the ongoing search for a truly global IoT ecosystem.

III. HYPERCAT

In early 2013, the UK's Technology Strategy Board invested in a project called the Internet of Things Ecosystem Demonstrator [15] to stimulate the development of an open IoT application and services ecosystem. In this project, eight industry led sub-projects were funded to deliver IoT clusters in the spring of 2014. Each cluster focused on different domains, for example airports, city transportation, smart homes, schools, highways, etc. One of the key goals of the project focused on interoperability, specifically on how the UK's IoT ecosystem could achieve interoperability and data availability between clusters in different domains.

Recognizing that full IoT interoperability is a large undertaking, the project focused on providing application developers with information about what data is available, what it represents, and how it is represented. To accomplish this, each cluster was tasked to create one or more IoT "hubs" that describe the devices they manage or represent to the web, and permits applications and services to interact with them. Each hub is then responsible for interacting with applications, and potentially, other hubs. Initially the project focused on supporting application developers with a specification for exposing diverse sets of IoT resources such as real time sensor data feeds, meta-data, and static asset datasets that describe "things". The outcome of this effort was HyperCat [14].

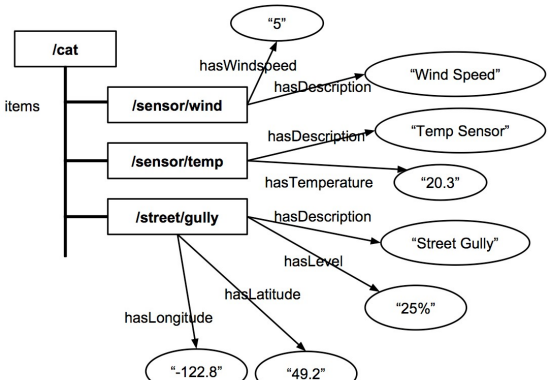

Fig. 2. HyperCat catalogues consist of a list of resources annotated with relationships and values.

HyperCat is a specification for a lightweight hypermedia catalog for querying and representing catalogs of resources (URIs) on the web. Exposed resources are described by a list of RDF-like triple statements to provide information about the format and semantics of the URI as illustrated in Figure 2. This enables applications to search for suitable resources and understand the data when they retrieve it. Because of its simplicity, developers can easily publish descriptions of the resources they expose; applications can easily query for the things they are interested in.

To do so, applications access a top-level catalogue that every HyperCat hub must expose. The catalogue itself is a resource representing an unordered list of items. Each item refers to a single URI, which may itself be another catalogue. An example of a simple catalogue is shown in Figure 3.

HyperCat specifies how to insert, update and delete catalogue items and provides a limited set of metadata relationships to describe catalogues and items. The relation *urn:X-sbiot:rels:hasDescription:en,* for example, is used to provide an English description of an item; *urn:X-sbiot:rels:isContentType* indicates the MIME type of the data associated with an item. The initial specification defines a

```
{
 "metadata":[
  { "rel":"urn:X-tsbiot:rels:isContentType",
    "val":"application/vnd.tsbiot.catalogue+json" },
  { "rel":"urn:X-tsbiot:rels:hasDescription:en",
    "val":"Bare catalogue" }
 ],
 "items":[
  { "href":"http://hub.com/resource1",
    "metadata":
    [
     { "rel":"urn:X-tsbiot:rels:hasDescription:en",
       "val":"The first resource" }
    ]
  },
  { "href":"http://hub.com/resource2",
    "metadata":
    [
     { "rel":"urn:X-tsbiot:rels:hasDescription:en",
       "val":"The second resource"}
    ]
  }
 ]
}
```

Fig. 3. Example HyperCat catalogue.

simple search capability where query parameters are used to query for a specific resource URI (*href* parameter), metadata relationships (*rel* parameter) and/or values (*val* parameter). If

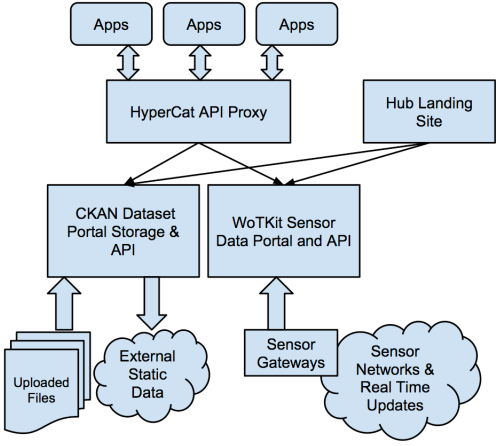

Fig. 4. High-level Smart Streets Hub Architecture

multiple parameters are supplied, the server is required to return the intersection of items that match all search parameters. Catalogues advertise whether they support search by using a *tsbiot:rels:supportsSearch* metadata relation. Finally, HyperCat specifies an optional security mechanism whereby all requests may be authenticated with a key presented using HTTP basic authentication, or in an *x-api-key* header. Details related to visibility and access control to catalogues were considered out of scope for the initial phase of the project.

A. *Smart Streets IoT Hub*

As developers of the Smart Streets IoT Hub, our focus was the Highways maintenance sector (a $6B sector in the UK), which gathered data from a variety of sources related to the UK's national and regional road network. Data included real-time road traffic, incidents that affected traffic, road works, flood and rain data, etc., all of which were made available via the Smart Streets IoT Hub[1].

A critical challenge we faced was the need to collect and manage a diverse set of existing data sources ranging from real time data on traffic flow or water levels in roadside drains, to soft real-time data such as roadwork schedules to relatively static data such as highways asset lists of signs, bridges, markings etc. in our IoT hub and provide a uniform API (via HyperCat) to this data.

The Hub itself was built from two core components (Fig. 4). Firstly, our own IoT platform called the Web of Things ToolKit (WoTKit) [3], [16], and secondly, an open source data management tool provided by the Comprehensive Knowledge Archive Network (CKAN) project [17], designed to support static data and metadata storage as illustrated in Figure 4.

IV. INTEROPERABILITY TOOLS

The CKAN system is a data management system and portal that allows data publishers like governments, companies and other organizations to make their data available to others. It makes it easy for data publishers to easily upload and publish new datasets containing one or more data resources, providing versioning and support for multiple formats. Datasets can be associated with organizations for access control. Grouping, tagging and metadata are supported to facilitate search. CKAN provides an API that allows developers to search for, download and, in some cases, query for data within relevant datasets. In Smart Streets, we used the CKAN system to store data sets that are static or do not change often (e.g. monthly or annually).

[1] Smart Streets Data Hub." https://smartstreets.sensetecnic.com/. Accessed: 28-Mar-2014.

The WoTKit, under development since 2009, is a web-centric IoT toolkit, focused on managing things that exhibit real-time behaviour. Running as a cloud service, its APIs offer developers a comprehensive set of IoT services making it easy to develop web applications and services for the IoT. Users create 'sensors' with the UI or API that represent 'things', can receive data from those things, and can send control commands. Like CKAN, sensors can be arranged into organizations for access control and can be grouped and tagged. Unlike CKAN, data is typically not uploaded as a single file, but on an ongoing basis, either periodically or when sensor values change, typically every few minutes.

Both CKAN and the WoTKit support API calls to view a 'catalogue' of resources (datasets and sensors), but the formats and APIs to access these catalogues are very different. To support interoperability, we needed to adapt the catalog APIs for both the WoTKit and CKAN to the HyperCat simple search parameters and semantics.

A. *The API Proxy*

To achieve this, we developed an API proxy architecture implemented as a web application in Python, using the Pyramid framework[2]. Like other web frameworks, Pyramid maps URLs to methods in control modules. In the API Proxy, when the corresponding URL (method) for a given catalog is requested, an object called *CatalogCreator* is instantiated. To support CKAN and the WoTKit and associated sub catalogues, we sub-classed this object for our implementations as illustrated in Figure 5.

B. *Search engine implementation*

To support flexible search, our initial CatalogCreator implementations leveraged the Apache Solr search platform[3] to both store and search catalogs (right side of Figure 5). We created data importer scripts that periodically called the CKAN and WoTKit APIs, and imported their data into solr 'cores' that could then be searched by the API Proxy. While this solution worked well for public sensors and data sets that did not change often, catalogue visibility and access control of private sensors and datasets by the underlying systems could not be supported without replicating the access control logic of the underlying systems. Moreover, if the catalog changed, the catalog exposed by the API Proxy would be out of date until the next catalogue import.

C. *Catalogue proxy implementation*

To address these issues, we created *ProxyCatalogCreator* sub-classes. These implementations translated HyperCat queries format to an appropriate API call to the underlying CKAN or WoTKit system, and then converted the response to the HyperCat format 'on the fly'. While the concept seemed straightforward, it raised a number of issues related to access control and security, query capability/semantic mismatch and catalogue scale.

1) *Unified access control*

To ensure that users could only view and access datasets or sensors they were permitted to, we needed to unify the user

[2] "Pylons Project Home."http://www.pylonsproject.org/. Accessed 1-Apr-2014

[3] "Apache Solr."https://lucene.apache.org/solr/. Accessed 1-Apr-2014

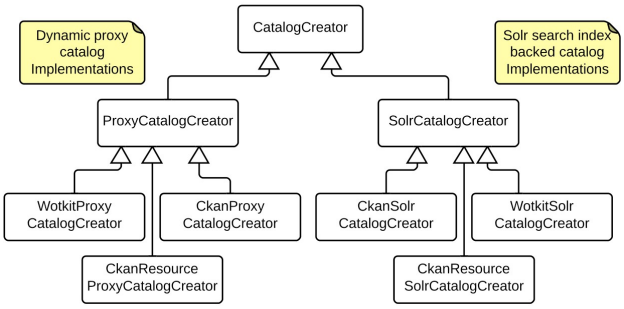

Fig. 5. CatalogCreator class diagram

accounts and access control mechanism used by both systems. The CKAN API uses a single developer key per user for access control, while the WoTKit API supports both basic authentication and OAuth 2. Like CKAN, HyperCat only required a single key for authentication and did not specify how the key is obtained or what it contains. To unify the systems under a single HyperCat key, we decided to modify the WoTKit to support CKAN authentication keys, associating this key with a WoTKit user and ensure the user credentials in both systems were kept in sync on both systems. Another approach would have been to maintain a mapping in the API Proxy to API keys or access tokens in the underlying systems. With unified access control, only the sensors or CKAN data sets visible to the user associated with the API key will be queried and returned by the API Proxy.

2) Query mismatch and filtering

Another issue was related to supporting catalogue queries. The initial query semantics for HyperCat, called 'simple search', allows users of the API to query for certain catalogue items by specifying whether a metadata relationship exists, and/or is set to a certain value. To perform a simple search, clients provide a query string specifying the specific item URI, relationships and/or values of the items they are interested in (href, rel and val parameters). All items with metadata that matches the query parameters must be returned.

While this sounds straightforward, because of limitations or the semantics of certain metadata in the underlying system, we weren't always able to search by the existence or value of certain metadata, nor did it always make sense to do so. For example, we included location as metadata using longitude and latitude relationships as the WoTKit maintains the location of fixed sensors. Since the WoTKit API did not provide a mechanism to find all sensors with a certain latitude value, just geo-queries in an area, we could not map the HyperCat simple search to the WoTKit search API directly. In the WoTKit, we provided a way to find sensors that contained certain text strings in the name, or description, but not to find sensors that matched these metadata values exactly. To address this, we extended the ProxyCatalogCreator implementation to support filtering a HyperCat response by query relationships and values after it was generated by the underlying system. While this worked, it meant that we had to retrieve more data than necessary, reducing the performance of a simple search.

3) Large catalogues

Since the WoTKit contained large amounts of real-time data, e.g. more than 40,000 gully/drain sensors from a given region, we needed a way to partition this large catalogue into manageable 'chunks' to avoid overwhelming clients of the HyperCat API. One approach we considered was to split the sensor catalogue into sub catalogues corresponding to 'pages' of a larger catalogue. Another was to partition our catalogue into sub-catalogues corresponding to the owner of datasets or sensors. On further consideration, we found that either approach would interfere with the ability to search the entire catalogue since HyperCat's simple search applies only to a single catalogue, not sub catalogues. Realizing this issue could not be addressed adequately with the current specification, we requested a change to support catalogue paging and new query parameters. These limited the number of items returned by a query on a large catalogue that co-exists with the HyperCat search query parameters.

Data interoperability: The Harvester

A secondary issue we faced was integrating a variety of heterogeneous data from a set of disparate sources ranging from lists of street fixtures (signs, lamps, barriers etc.) to collected statistics on road repair performance and impact. To address this, we developed an additional tool called the *Harvester*, loosely based on the CKAN Harvester plug-in used for federating CKAN open data portals. Like data warehouse Extract Transform and Load (ETL) tools, the Harvester integrates sources of data into a common 'web of things' repository. Rather than simply extracting data from databases, the Harvester extends that capability to also extract virtual sensor data buried in web pages, XML data feeds and other web formats. It normalizes this data and uploads it into the WoTKit for easy access by developers. Today the Harvester loads data into several instances of the WoTKit platform using its REST API.

V. RELATED WORK

Aggregating many things into cloud-based IoT hubs is a growing trend. Some are product or market-centric, others focus on specific domains. Interoperability is also being addressed at different levels by various industry and standards groups.

Many of today's Internet-connected products use web-based services for remote control and monitoring employing mobile phone applications and web browsers. The 'Nest' thermostat [18], for example, is connected to a cloud service using home wifi networks ,permitting users to manage their home heating with their mobile phones and the web. The Koubachi system [19] connects consumer plant sensors to the web, allowing owners to monitor and improve their plant health with alerts and graphs. It supports a RESTful API for developers to access this data from other applications. The Smart Things platform provides an API, a programming language and web based IDE for creating home automation applications [20].

Large IoT hub and platform vendors provide a degree of interoperability by providing a thing-agnostic model and API to integrate things across a wide variety of domains. These vendors aim to create a network effect where the hub becomes more valuable as more users and their things are connected. By doing so, vendors hope to establish their hubs as de-facto standards for web of things interoperability. Our own work, the WoTKit [3], as well as Xively(formerly Cosm and Pachube [8]), aggregate collections of data streams called feeds to store information about sensors and the data they emit over time. Similarly, ThingSpeak [7] supports a data model of channels similar to Xively and WoTKit feeds. All three include applications for processing, visualization and integration, and offer the ability to find and share sensors and data, allowing others to take advantage of the integration work of others. The

Each of these platforms offer a 'hub' model to provide a repository for Things (data and metadata) and a set of APIs for accessing and using Things.

The Internet of Things Architecture project (IoT-A) is proposing an architectural reference model for IoT interoperability together with key components of the future IoT to enable search, discovery and interaction as one coherent network [21].

The IETF community has been involved in foundational IoT technologies such as IPv6 and the Constrained Application Protocol (CoAP), focusing on getting constrained devices and sensor networks connected to the Internet [11], [22]. Similarly, the IEEE has several protocol standards that form the foundation of the IoT and provid connectivity between things and the Internet [10].

The Open Geospatial Consortium (OGC) and others [23] have recognized the need for coordinating systems to make it easier for applications to discover and access a wide variety of sensors independent of connectivity and data types. The OGC Sensor Web Enablement (SWE) framework defines a standard set of web service interfaces making it easier to share sensor data. A new working group called Sensor Web Interface for the IoT [12] aims to link emerging web of things toolkits and platforms to the OGC SWE standards.

While these efforts are moving the IoT toward greater interoperability, some, such as the IETF and IEEE, deal primarily with connecting things to the Internet and the web. They specify only the core networking infrastructure and protocols needed, not cross-domain hub-to-hub or hub-to-application catalogue and data interoperability as outlined here. More ambitious standardization efforts such as the IoT-A project [21], although offering a comprehensive approach to interoperability, are hampered by their scale, and may be attempting to lock down aspects of the IoT ecosystem while it is still rapidly evolving. Developers and vendors however, may favour less complex approaches that address requirements as they emerge.

VI. EXPERIENCES AND LESSONS

The Smart Streets IoT hub has been in operation for approximately 8 months and currently manages 64,000 time-series sensor feeds as well as a wide variety of static datasets. It includes a diverse set of both open and private data about transportation, road traffic and highways, ranging from real-time traffic data to road asset condition, planned roadworks, air quality, weather and flooding information. These data sources have been pushed into the hub either via tools such as the Harvestor, by end users uploading data sets, or from physical devices that explicitly send information to the Hub via its APIs.

During the course of the project, we have built a variety of IoT applications that use our Smart Streets hub as well as data from other hubs. The *Catalogue Explorer* connects to other hubs' HyperCat catalogues to display and view things available on other hubs. We used JavaScript visualization frameworks to build a *roadworks mashup* that uses roadwork and sign data to display the location and severity of accidents. A *Traffic Data Explorer* application displays roadworks against traffic flow and the relationship of traffic flow to delays,.A gully visualization mashup that explores the correlation between road works and gully silt levels over time. At a recent hackathon, 50+ participants from Switzerland, Germany and the UK developed a series of apps. Over a two day period generated more than 300K Hub API calls transferring over 9 GB of data.

Making a diverse set of IoT resources discoverable on the web by using a common catalogue was the initial focus of the UK IoT project. Even with the limited scope of the HyperCat specification (catalogs, simple security, simple search), we found that achieving a level of meaningful interoperability using our API proxy was difficult. Some of the challenges included the need to resolve different access control mechanisms, different query semantics, and dealing with the size of certain large catalogues.

Related to security and access control, we found that it is not enough to control *access* to the data for thing resources hosted on an IoT hub. It is also critical to control the visibility or knowledge that a resource is even available, limiting or exposing its visibility in a catalogue depending on its ownership and the user, or application accessing the catalogue API. To ensure these controls were reflected in the exposed catalogue, we could not simply replicate the catalogue of underlying systems; we needed to access these catalogues directly using the credentials of the requesting client to ensure only resources visible to that client were retrieved as described in section IV.

Overall we found that although the scope of HyperCat was appropriate for this initial project, it became obvious once several of the hubs came on-line that the groups will need to work closely toward standardizing catalogue item semantics, agreeing on the definition of certain meta data relationship fields and on how they relate to search, access control and data formats.

We believe that the use of ETL tools such as the Harvester is a good interim step towards addressing data interoperability,. The Harvester framework allowed us to aggregate a diverse set of data sources into a single hub that exposed a consistent API for developers, allowing developers to focus more on their applications rather than worry about the location and formats of the data they needed on the web. During our efforts to populate our hub with interesting data sets, we have found that there is an abundance of thing and sensor data available on the web today, buried in various web sites. In many cases these sites focus on providing end users with information and do not consider the value of their data as part of an IoT ecosystem, in which application developers can combine their data with that of others. We believe this will change over time as data suppliers standardize data formats and the metatdata used to describe these representations (e.g. MIME types). Based on our experience with HyperCat,our hub containing both CKAN and WoTKit data feeds and the diverse data already available on the web, we believe that it may be more practical to agree on the metadata (relationships and values) used to describe domain-specific data formats rather than to agree on one format. Using a catalogue like HyperCat, with a flexible metadata facility for describing things, will allow application developers to decide whether they are capable of consuming the data exposed by a given resource.

VII. CONCLUSIONS

Interoperability in the IoT is critical to achieving the potential of the widest variety of applications and services that can interact with objects in our physical world. Research and development to date have shown that a web-centric approach is a critical first step in achieving this vision. With the introduction of IoT hubs that aggregate IoT resources using web protocols, application developers can access individual hub-hosted physical resources such as environmental sensors, home automation equipment, home appliances and other things in a uniform manner. The challenge of interoperability then becomes one of unifying the presentation of hub catalogues and data formats. While we believe standards groups will eventually be able to provide interoperable specifications for WoT hubs, the tools presented here allow hub developers to begin to address interoperability while the requirements for such specifications become clearer and agreement is reached between academic and industrial practitioners in the IoT community.

ACKNOWLEDGEMENTS

We are indebted to our colleagues in the SmartStreets IoT project team, especially those at In Touch Ltd. and Lancaster University. The HyperCat work is the result of collaboration by the 8 IoT hub projects who participated in the interoperability working group. Partial funding for this work was provided by the TSB and NSERC.

REFERENCES

[1] A. Whitmore, A. Agarwal, and L. D. Xu, "The Internet of Things—A survey of topics and trends," *Information Systems Frontiers*, pp. 1–14.

[2] D. Guinard, V. Trifa, F. Mattern, and E. Wilde, "From the Internet of Things to the Web of Things: Resource Oriented Architecture and Best Practices," in *Architecting the Internet of Things*, D. Uckelmann, M. Harrison, and F. Michahelles, Eds. New York Dordrecht Heidelberg London: Springer, 2011, pp. 97–129.

[3] M. Blackstock and R. Lea, "IoT mashups with the WoTKit," in *Internet of Things (IOT), 2012 3rd International Conference on the*, Wuxi, China, 2012, pp. 159–166.

[4] D. Guinard, V. Trifa, T. Pham, and O. Liechti, "Towards physical mashups in the web of things," in *Proceedings of the 6th international conference on Networked sensing systems*, Pittsburgh, Pennsylvania, USA, 2009, pp. 196–199.

[5] M. Blackstock, R. Lea, and A. Friday, "Uniting online social networks with places and things," in *Workshop on the Web of Things (WoT 2011)*, San Francisco, CA, USA, 2011, pp. 5:1–5:6.

[6] D. Guinard, M. Fischer, and V. Trifa, "Sharing using social networks in a composable Web of Things," in *2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, 2010, pp. 702–707.

[7] "The Internet of Things - ThingSpeak." [Online]. Available: https://thingspeak.com/. [Accessed: 29-Jan-2013].

[8] "Xively by LogMeIn." [Online]. Available: https://xively.com/. [Accessed: 01-Apr-2014].

[9] M. Blackstock and R. Lea, "Toward Interoperability in a Web of Things," in *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*, New York, NY, USA, 2013, pp. 1565–1574.

[10] "IEEE-SA - Internet of Things Related Standards." [Online]. Available: http://standards.ieee.org/innovate/iot/stds.html. [Accessed: 28-May-2013].

[11] I. Ishaq, D. Carels, G. Teklemariam, J. Hoebeke, F. Abeele, E. Poorter, I. Moerman, and P. Demeester, "IETF Standardization in the Field of the Internet of Things (IoT): A Survey," *Journal of Sensor and Actuator Networks*, vol. 2, no. 2, pp. 235–287, Apr. 2013.

[12] "Sensor Web Interface for IoT SWG | OGC(R)." [Online]. Available: http://www.opengeospatial.org/projects/groups/sweiotswg. [Accessed: 28-May-2013].

[13] "W3C Web of Things Community Group." [Online]. Available: http://www.w3.org/community/wot/. [Accessed: 26-Mar-2014].

[14] "HyperCat Specification," *HyperCat Specification*, Dec-2013. [Online]. Available: http://wiki.1248.io/doku.php?id=hypercat. [Accessed: 26-Mar-2014].

[15] "Internet of Things Ecosystem Demonstrator Overview." [Online]. Available: https://connect.innovateuk.org/web/internet-of-things-ecosystem-demonstrator. [Accessed: 28-May-2013].

[16] M. Blackstock and R. Lea, "WoTKit: A Lightweight Toolkit for the Web of Things," in *Proceedings of the Third International Workshop on the Web of Things*, New York, NY, USA, 2012, pp. 3:1–3:6.

[17] "CKAN The Open Source Data Portal Software." [Online]. Available: http://ckan.org/.

[18] "Nest | The Learning Thermostat | Home." [Online]. Available: http://nest.com/ca/. [Accessed: 28-May-2013].

[19] "Koubachi - Interactive Plant Care." [Online]. Available: http://www.koubachi.com/main?locale=en. [Accessed: 26-Mar-2014].

[20] "Smart Things." [Online]. Available: http://smartthings.com/. [Accessed: 26-Mar-2014].

[21] "Internet of Things - Architecture — IOT-A: Internet of Things Architecture." [Online]. Available: http://www.iot-a.eu/public. [Accessed: 28-May-2013].

[22] A. P. Castellani, M. Gheda, N. Bui, M. Rossi, and M. Zorzi, "Web Services for the Internet of Things through CoAP and EXI," in *2011 IEEE International Conference on Communications Workshops (ICC)*, 2011, pp. 1–6.

[23] M. Botts and A. Robbin, "Bringing the Sensor Web together," *Géosciences*, vol. 2007, no. 6, pp. 46–53, Oct. 2007.