SOXFire: A Universal Sensor Network System for Sharing Social Big Sensor Data in Smart Cities

Takuro Yonezawa, Tomotaka Ito, Jin Nakazawa, Hideyuki Tokuda
Keio University
5322 Endo, Fujisawa, Kanagawa, Japan
{takuro, tomotaka, jin, hxt}@ht.sfc.keio.ac.jp

ABSTRACT

We introduce SOXFire, a multi-community city-wide sensor network for sharing social big sensor data in smart cities. The goal of SOXFire is to provide practical distributed and federated infrastructure for IoT sensor data sharing among various users/organizations in a way that is scalable, extensible, easy to use and secure with preserving privacy. SOXFire supports not only access to physical IoT sensors but also crowd sensing and SNS/Web sensing where city employees, citizen and WEB developers contributed in a different ways but unified APIs. In this paper, we outline the concept of SOXFire and present it's design and implementation with tangible libraries and tools. In addition, we discuss capability and usability of SOXFire based on our experiments in smart city project in Japan.

CCS Concepts

•Software and its engineering → Publish-subscribe / event-based architectures;

Keywords

Smart City; IoT; Requirements; Architecture

1. INTRODUCTION

The world is facing several challenges that must be dealt with in the coming years such as efficient energy management, need for economic growth, security and quality of life of its habitants. The increasing concentration of the world population into urban areas put the cities in the center of the preoccupations. To tackle these problems, many researchers are focusing the way to make city more smart. Smart city, though there exist several definitions for it, is defined as a complex ecosystem characterized by the intensive use of information and communication technologies, aiming at making the cities more efficiency, more attractive, more sustainable and a unique place for innovation and entrepreneurship.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

WOODSTOCK '97 El Paso, Texas USA

© 2016 ACM. ISBN 123-4567-24-567/08/06…$15.00

DOI: 10.475/123_4

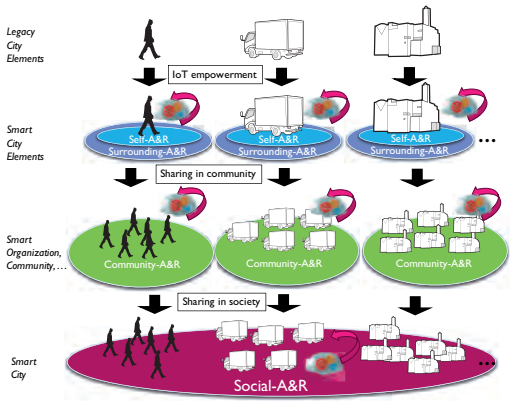

Figure 1: Layering model of smart city

The word of smartness means ability to think and respond quickly and effectively - to be responsive to all going around, fast to analyze, reason, plan and make decisions, and fast to react with desirable effects. Thus, smart city must have ability of awareness and responsiveness to dynamic change of the city. Of course, since city is composed of various elements such as people, vehicle and buildings, smart city also must be composed by smart people, smart vehicle, smart buildings and their composition such as smart company, community and so on. Figure 1 depicts the layering model of smart city. The topmost one is the Legacy City Elements layer, namely the physical world elements. The elements are augmented by IoT technologies to empower the elements with awareness and responsiveness. For example, a car can be augmented by sensors attached to it to acquire self-/surrounding awareness and responsiveness. The data generated by the smart city elements are exchanged among them to better serve the level of community or organization, and more integrated in a cloud system to generate social awareness and responsiveness at the city-wide level. This model shows that acquired IoT data at city elements is consumed at elements-level, but at the same time useful data (or data after be analyzed) should be shared and consumed at community-level and city-level. We call the data, which has usefulness and sociality to be shared, as social big sensor data.

In this paper, we present SOXFire, a universal sensor network system for sharing social big sensor data to realize sensor data ecosystem in smart cities. Social big sensor data is required to be shared various level such as city elements, community-level and city-level. This means that not only heterogeneous sensors but also heterogeneous users at differ-

ent organizations have to be concerned. Therefore, a system for sharing social big sensor data have to cope with various sensors, and share the data with various users/organizations at different access rules. In addition, the system have to be managed and operated easily in each level, and also have to provide scalability and extensibility for tons of sensor data from cities. SOXFire is designed to solve this issue by leveraging XMPP Internet messaging protocol to provide federated server functionality as well as publish-subscribe communication. SOXFire is based on Sensor Andrew[11] which is also XMPP-based large-scale campus-wide sensing system, however, we extended the concept of Sensor Andrew for smart city by introducing several important features to cope with more variety of sensors and users.

The paper is structured as following. Firstly, we present requirements to the system for sharing social big sensor data. Then, SOXFire is introduced with it's design and implementation. We also provide tangible tools for SOXFire. Lastly, we describe actual on-going smart city projects based on our proposed system. Lastly, we present related work and conclude the paper.

2. DESIGN GOAL

Towards smart city era, types of sensors and their users will be more diverse and increased. Usually, sensor network is operated in closed environment and system - types of sensors are limited, and sensor deployers and consumers are usually in same organization. However, since smart city is a complex system of systems, tons of heterogeneous sensor data must be leveraged by thousands of heterogenous users. Thus, it is necessary to design a large-scale sensor network system to share such social big sensor data. We adopt the following requirements for the sensor network system in smart city.

1) **Coping with heterogeneous sensors:** Not only physical IoT sensors but also crowd sensing and WEB sensing are key sensing techniques in smart cities. Therefore, the system have to cope with those various ways of sensing.

2) **Compatible with easy and secure access:** Social big sensor data requires both features of openness and secureness. While some data can be accessed by anyone, some data must requires strict access control.

3) **Retrieving sensor data in different ways:** Sensor data usage pattern is different by different applications. An application may require only single sensor data at current time, or other application may require continuous sensor data stream. Therefore, the data should be retrieved by both pull-oriented and push-oriented to fulfill various requirements of applications' use cases.

4) **Ease of management, operation and use:** Easiness of system management and use needs to be considered for practical system. This should also requires a easy process of registering and discovering sensors of interests. In addition, simple ways of development are also required for developers of applications or their own sub systems.

5) **Scalability and Extensibility:** The system should have scalability to manage tons of sensor data. In addition, it should provide extensibility for system administrators/developers to extend the system for their purpose.

6) **Federation of multi-community system:** As shown in Figure 1, users in each layer use their sensor data for their own applications in distributed manner. At the same time, shareable data should be opened for other community, or wider social layer. Thus, this distributed and multi-community system have to be federated for smart cities ecosystem.

These are fundamental design requirements for a universal sensor network system in smart cities, however, it would be infeasible to design and implement the system which fulfills the requirements from scratch. Especially, there are several matured protocols for IoT communication such as MQTT, CoAP and XMPP. These protocols are already widely used in active developers' communities, and the communites provides several useful protocol implementation and API libraries for several programming language. Therefore, we focus to extend existing technology to fulfill the design goal of universal sensor network in smart cities.

3. SOXFIRE

3.1 Background technology: XMPP

To fulfill the requirements described in previous section, we chose to leverage the eXtensible Messaging and Presence Protocol (XMPP). XMPP is an open XML-formatted Internet protocol traditionally used for chat communications. XMPP provides several features such as user accounting and authentication, flexible access control mechanism, publish-subscribe(pubsub) event model, scalable clustering and federation features, and so on. In addition, XMPP uses XML so that it can provide extensibility for developers to define their own scheme by themselves. There are several other IoT protocols such as MQTT and CoAP and we could actually leverage them. Nevertheless, each protocols can be the best choice for specific purposes. However, we consider XMPP is the one of best practical protocols to fulfill design requirements in terms of social big sensor data sharing currently. Actually, there have been several on-going projects to leverage XMPP for IoT communication. Of those, Sensor-Over-XMPP(SOX) specification proposed in Sensor Andrew project leverages pubsub model for sensor data sharing, which inspires us and is actually basement of our system. However, these works focus rather for physical IoT sensors with specific use cases which cannot fit to the requirements described in the previous section. Our contribution is to leverage XMPP protocol for sharing social big sensor data among various layers of users in smart cities.

3.2 Virtual sensor model in SOX

As mentioned before, our sensor model extends SOX specification proposed in Sensor Andrew project. In SOX specification, physical sensors are expressed as corresponded virtual sensors as pubsub event nodes. Using pubsub model is an ideal way for massive distribution of IoT data, as same as in MQTT. However, one of advantages of SOX is that it can express sensor's meta information in nature. This enables users to understand/discover required sensors - this is

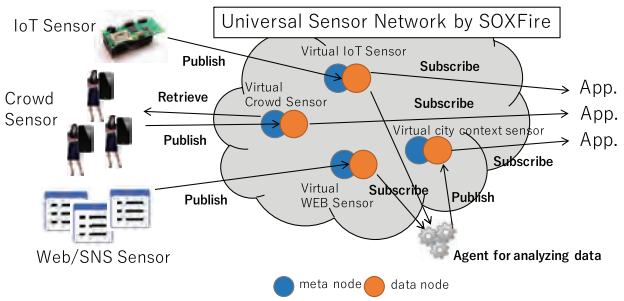

Figure 2: Virtual sensor model in SOXFire

important feature for sharing social big sensor data among multi-community environment. In SOX, a virtual sensor corresponded to a physical sensor is associate to two pubsub nodes, named *SensorName*_meta and *SensorName*_data. Figure 2 shows overview of relationship between real-world sensors and virtual sensors in SOX. PubSub node which has postfix _meta is used for publishing meta information such as sensor device name (e.g., mySensor), device types (e.g., outdoor weather), sensors mounted on the device (e.g., temperature and humidity), their units (e.g., celsius and percent) and so on. PubSub node which has postfix _data is used for publishing actual sensor data (e.g., 24 and 76). Our SOX libraries, which will be described in next section, hides this paring of _meta and _data, so users/developers can easily access to both meta information and actual data without taking care of the difference of _meta and _data nodes.

We can set flexible access control for pubsub node such as 'open', 'authorize' or 'whitelist' for subscribing the node, and 'open', 'publishers', 'subscribers' for publishing data to the node. By using the access control, we can provide following different sensor pattern in SOX.

Pattern 1 Publisher:Virtual Sensor = 1 : 1

This model means that a physical entity is associated to a virtual sensor node. This is basic model for IoT sensor node. For example, suppose that a sensor node which has accelerometer and temperature sensor is only allowed to publish those sensor data to a corresponded virtual sensor. The data published to virtual sensor is distributed to any users. In this case, we should set publishing model as 'publishers', especially only for the owner of the sensor node, and set access model as 'open' for anyone (see 1 in Figure 3).

Pattern 2 Publisher:Virtual Sensor = n : 1

This model means that anyone (or allowed users) can act as sensor data publisher to a virtual sensor node. This is basic model for crowd sensing. In this case, we should set both publishing model and access model as 'open' (see 2 in Figure 3).

We extended original SOX specification to cope with not only IoT sensors but also crowd sensing. Crowd sensing usually ask users to report certain findings in cities (e.g., road damage reporting with it's picture, etc.). As meta data expression for crowd sensing, we added several sensor types such as 'participatory' for crowd sensing, and provide related units such as 'singleChoice', 'multipleChoice', 'freeFormText' or 'image'. This meta data also can be used to generate reporting interface for publishers [12].

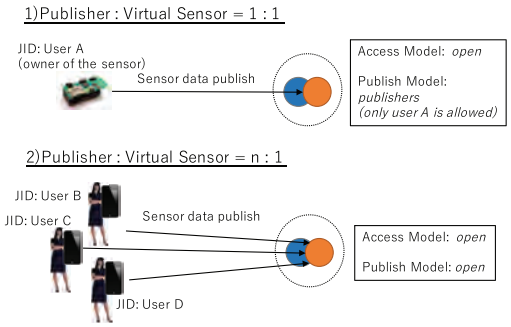

Figure 3: Different sensor data publishing pattern

3.3 Requirements fulfillment

In addition to flexibility of SOX's virtual sensor model for coping with heterogeneous sensors, SOXFire satisfies design requirements in the following ways by leveraging XMPP technology:

- XMPP's PubSub function offers not only pubsub-based access but also explicit data retrieving. Users can retrieve data freely which is stored in database of SOXFire. SOXFire is designed to persist meta information, and enable node owners to decide the number of historical sensor data to be kept.
- XMPP provides secure features such as user/group authorization, authentication, access control and data encryption. On the contrary, XMPP does not suppose node subscription by anonymous users. For the data to be allowed for sharing with anyone, SOXFire provides anonymous subscription/data retrieving. This allows end-users to access open sensor data without any user registration.
- XMPP can utilize clustering for scalability and robustness. Also, XMPP has server-to-server federation functionality. This allows basic communications between servers, however, subscription over federation is not specified in XMPP. SOXFire enables users to subscribes nodes on federated servers.

As described above, XMPP has its limitations in our requirements. SOXFire is a modified implementation of XMPP server based on an open source software to solve these problems. In addition, we provide several practical libraries for application programmers.

4. IMPLEMENTATION

4.1 SOXFire server

We explored several XMPP server implementation such as ejabberd, Openfire, tigase and so on. Of those, we chose to use and modify Openfire, a open source XMPP server software (under open source apache license) implemented by Java. Openfire has been developed originally from 2002, and was wholly handed to the community. It provides easy install, easy administration and several useful plugins such as clustering. Though Openfire offers basic functionality as XMPP server, there are several limitations for satisfying our requirements. To solve these limitations, we modify following points of Openfire source code as SOXFire.

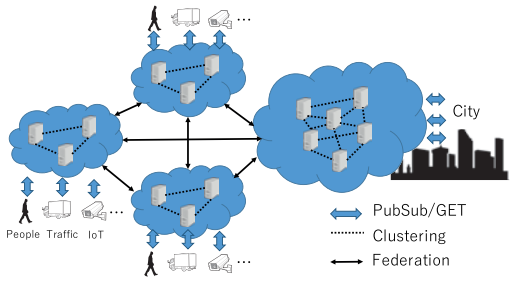

Figure 4: Scalable and federated SOXFire architecture

- Enabling anonymous subscription: In Openfire, subscriptions are persistent and the information is stored in the database, subscription is only allowed for registered users. However, as mentioned in requirements, social big sensor data should be easily accessed by anyone. Thus, SOXFire allows anonymous subscription.
- Enabling subscription over server-to-server federation: Openfire provides server-to-server federation functionality. However, it does not offer subscription over federation. SOXFire enables users to subscribe node on federated SOXFire server. With clustering functionality and federation functionality, SOXFire architecture provides scalable, distributed and federated sensor network architecture for smart cities (see Figure4).
- Optimizing database transaction: XMPP core functionality in the server side causes a lot of database transactions for data caching, meta-data storage/retrieval, and keeping connections from anonymous users. We improve usage of database by monitoring connection of users including anonymous users and server-to-server federation.

Our implementation of SOXFire is also opened under open source apache license. Therefore, developers can freely download, install, use and also modify SOXFire. We are now operating six SOXFire servers (including operation by a third party) for different community and usage purpose, and they are federated each other to access sharable sensor data easily.

4.2 SOXFire APIs

We also provide complementary client-side APIs for different programming languages including Java, Javascript, Objective-C and python. SOXFire libraries hides complex XMPP communication and existence of two types of nodes _meta and _data, and enables users to access virtual sensors in simple ways. Followings are usage examples of Java SOXFire API.

- Connecting to a SOXFire server

```
SoxConnection con =
        new SoxConnection("server-name.org");
```

- Getting virtual sensors list

```
//get the list on connected server
List<String> nodeList = con.getAllDeviceList();

//get the list on federated server
List<String> nodeList2 =
  con.getAllDeviceList("other-server.org");
```

- Associating a virtual sensor to SoxDevice object

```
//associate to virtual sensor on connected server
SoxDevice sd = new SoxDevice(con, "sensor_name");

//associate to virtual sensor on federated server
SoxDevice sd2 =
        new SoxDevice(con, "sensor_name",
                      "other-server.org");
```

- Getting meta data information from _meta pubsub node

```
Device deviceInfo = sd.getDevice();
```

- Retrieving last sensor data from _data pubsub node

```
Data data = sd.getLastPublishData();
```

- Subscribing the sensor and handling data to be published

```
sd.subscribe();
sd.addSoxEventListener(this);
...
public void handlePublishedSoxEvent(SoxEvent e){
        List<TransducerValue> values =
        e.getTransducerValues();
        //then write processing of the data
}
```

Those APIs are object oriented, and share almost the same class hierarchy among different programming languages, so various kinds of applications can be developed easily on, for example, smartphones (both iOS and android), web pages and PCs.

4.3 Tangible tools

In addition to SOXFire server implementation and client-side APIs, we developed several tools for validating usefulness and capability of SOXFire. These tools are also open via our web page [1].

4.3.1 WEB Sensorizer

WEB sensorizer[9] is a novel tool for acquiring real-world data in a simple yet extensible way. Its major idea is to put "virtual" sensors on web pages that contain meaningful values sensed from the real world. Figure 5 shows such web pages that contains air quality information sensed in corresponding cities. The numbers shown in these pages are updated periodically, and the past numbers becomes unaccessible since they are stored deep in a database. Virtual sensors put on these pages periodically transmit the numbers, which are scraped from them. Since there are a number of web pages that contain real world data, and also deploying a virtual sensor needs just a few steps of GUI manipulation, virtual sensing can generate a huge amount of data that help understand the real world. On the system's aspect, our virtual sensing technique is a set of the following components.

[1] http://sox.ht.sfc.keio.ac.jp

Figure 5: Air Quality Web Site in Europe and Japan

Authoring Tool This client-side tool is an extension module of the Chrome browser that enables browser users to deploy virtual sensors on almost arbitrary elements on a web page.

Probe Probe is the server-side program that inputs a virtual sensor definition, which includes the URL of a web page and the target elements' XPath, and periodically scrapes the element values from the page. It also has functionality to explore similar structure's WEB pages and sensorize the page automatically. Probe uses Java version of SOXFire API.

The data scraped from web pages is published to SOXFire. Then, the data are transmitted to their subscribers via XMPP protocol. So far we have sensorized more than 400 thousands of WEB pages and being generating more than 20GB/day sensor data stream via SOXFire.

4.3.2 *SOX Dashboard*

SOX dashboard is a widget and map-based visualization software which allows users to create their own dashboard webpages. The application is implemented with JavaScript version of SOXFire API. Figure 6 (1) shows main menu of dashboard. Users can add new dashboard just inputting dashboard name. Figure 6 (2) shows an example of Fujisawa (a city of Japan) dashboard. The dashboard has 6 widgets - city monitoring camera widget, weather widget, participatory sensing widget, 'what day is today' widget, weather forecast widget and Twitter's popular image widget. After moving this dashboard web page, JavaScript SOX library starts to subscribe corresponded virtual sensors as anonymous user. WEB page is viewed by many people, anonymous subscription enables them to access the real-time data without any user registration. As shown in the figure, the widget can contain many kinds of information such as text-based data, image data and map data. These data comes from SOXFire. To add new widgets, users just click on the upper icon, then an interface will appear (see Figure 6 (3)). It is very easy to add widgets, just by selecting sensor node name (a search function is also available), deciding the panel type from text, image or map, associating sensor to panel. For map visualization, we used Google maps.

5. FIELD TRIALS

We also deploy and operate SOXFire for sharing various IoT sensing and crowd sensing in Fujisawa, Japan. We have interviewed city officers of Fujisawa city to find problems that affects residents and sightseers. As challenge to tackle those problems, we present two of field trials ongoing in Fujisawa city.

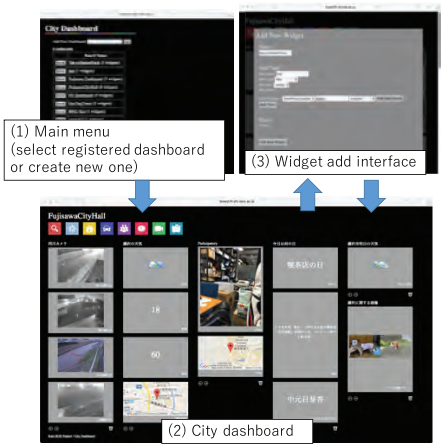

Figure 6: SOX Dashboard

5.1 Automotive sensing

It is known that a number of air pollutants, such as PM2.5 and NOx, affects human health. To help people know how much the air around each of them, there needs to be enough amount of information that is fined-grained enough. However, not only in Fujisawa, but also in other cities in Japan, there are a limited number of air pollutant monitoring stations in each city, just five in case of Fujisawa. Due to this, local residents and the sightseers are unable to be aware of quality of the air they breath.

In order to collect fined-grained environmental information from all around the city, and also to enable city officers to be aware of the location of garbage trucks, we have attached IoT sensors (e.g., temperature, UV, PM2.5, NOx and so on) almost all the garbage trucks operated by Fujisawa city[5]. Collected data is published to SOXFire in 100Hz frequency from each car, and distributed to application which visualizes environmental data.

5.2 Expert crowd sensing

In general, various kinds of resources are operated in a city. They include transportations, such as buses and taxies, official cars, parking lots, and so on. It is crucial to be aware of their real-time status to optimize city officers' dicision making. Their locations, for example, help officers to conduct an initial response to an enormous earthquake that is expected to hit the area containing Fujisawa within decades. However, such a real-time information is not collected.

In order to collect real-time city resources status and city anomalies, we have sensorized city officers in Fujisawa city. Fujisawa city has about 3,500 officers. Some of them are working outside of city hall, such as garbage collection, fire fighting, and teaching at public schools. The form of sensing that leverages city officers as human sensors is what we call *Expert Crowd Sensing*. City officers are expected to have more domestic knowledge of the city where they work than ordinary residents. For example, officers working on garbage collection travel through the area of responsibility everyday, thus know the whole area in detail. Consequently expert crowd sensing enables us to acquire more accurate human sensing data with more sensitivity. To realize the expert crowd sensing, we develop client application for sensing, and a web application for visualizing data based on SOX-

Figure 7: Screenshot of client/viewer application for expert crowd sensing

Fire. Since types of reporting tasks are different according to the officers' job sections, suitable types are associated to each job section. Users publish sensor data as following process using the client application: (1) select job section, (2) select report type, (3) take a picture and (4) specify a label and a comment and publish the data (see Figure 7 (1)). Then, reported data are visualized by the web viewer application (see Figure 7 (2)). Users can select each report by table-based interface or map-based interface. Then, details of report is shown in the viewer.

6. RELATED WORK

The smart city concept has implied the appearance of different research initiatives related to: i) sensor (and actuator) middleware architectures[1, 6, 3], ii) provision of tools and modules for implementing applications and services on top of these testbeds[2, 10], iii) development of services and applications within real urban environments[13]. These initiatives tackles fundamental architecture for smart city, however, mainly they are focusing on limited city resources such as IoT or open data in more centered manner. Our architecture can manage various city entities and manage them in distributed manner. Also, we provide complementary tools for crowd sensing and WEB sensing on proposed SOXFire system. On the contrary, our architecture currently does not support open data such as CKAN[2]. Thus, it is necessary to explore a way for more integration with various city data with SOXFire architecture. CityHub[7] proposes cloud-based smart city hubs which provides integration and interoperability of open data and sensor data, and enables users to access the data via REST API. Our architecture is more edge and cloud integration but it can cooperate with these research.

Nevertheless, our architecture still has several limitations. Firstly, our system is only focusing of data communication and distribution, not data analysis. Integrating online data analytics [4, 8]must be one challenge as future work. Secondly, our sensor discovery mechanism is still naive implementation. It is necessary to provide more efficient sensor discovery and leverage mechanism, such as context-aware search or interest-based data distribution.

7. CONCLUSION

We present SOXFire, a universal sensor network system for sharing social big sensor data in smart cities. SOXFire is designed to cope with heterogenous sensors with keeping ease of use, scalability, federation and extensibility. We modified implementation of XMPP server for meet the requirements, and also provided a set of useful tools such as client APIs, visualization software or WEB sensing mechanism. Based on our initial experiments with SOXFire in Fujisawa city, we consider that our architecture takes a great role for sensor data sharing from community-level to city-level environment.

[2]http://ckan.org

8. REFERENCES

[1] Butler project. http://www.iot-butler.eu/.
[2] Citysdk. http://www.citysdk.eu.
[3] Internet of things - architecture. http://www.iot-a.eu.
[4] Node-red. http://nodered.org.
[5] Chen, Y., Nakazawa, J., Yonezawa, T., Kawesaki, T., and Tokuda, H. Cruisers: A public automotive sensing platform for smart cities. In *36th IEEE International Conference on Distributed Computing Systems, ICDCS 2016, Nara, Japan, June 27-30, 2016*, IEEE (2016), 767–768.
[6] Giaffreda, R. *iCore: A Cognitive Management Framework for the Internet of Things*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, 350–352.
[7] Lea, R., and Blackstock, M. City hub: A cloud-based iot platform for smart cities. In *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on* (Dec 2014), 799–804.
[8] Li, F., Voegler, M., Claessens, M., and Dustdar, S. Efficient and scalable iot service delivery on cloud. In *Proceedings of the 2013 IEEE Sixth International Conference on Cloud Computing*, CLOUD '13, IEEE Computer Society (Washington, DC, USA, 2013), 740–747.
[9] Nakazawa, J., Tokuda, H., and Yonezawa, T. Sensorizer: An architecture for regenerating cyber physical data streams from the web. In *Adjunct Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers*, UbiComp/ISWC'15 Adjunct, ACM (New York, NY, USA, 2015), 1599–1606.
[10] Pfisterer, D., Romer, K., Bimschas, D., Kleine, O., Mietz, R., Truong, C., Hasemann, H., Kroller, A., Pagel, M., Hauswirth, M., et al. Spitfire: toward a semantic web of things. *Communications Magazine, IEEE 49*, 11 (2011), 40–48.
[11] Rowe, A., Berges, M., Bhatia, G., Goldman, E., Rajkumar, R., Jr., J. H. G., Moura, J. M. F., and Soibelman, L. Sensor andrew: Large-scale campus-wide sensing and actuation. *IBM Journal of Research and Development 55*, 1 (2011), 6.
[12] Sakamura, M., Ito, T., Tokuda, H., Yonezawa, T., and Nakazawa, J. Minaqn: Web-based participatory sensing platform for citizen-centric urban development. In *Adjunct Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers*, UbiComp/ISWC'15 Adjunct, ACM (New York, NY, USA, 2015), 1607–1614.
[13] Sanchez, L., MuÃśoz, L., Galache, J. A., Sotres, P., Santana, J. R., Gutierrez, V., Ramdhany, R., Gluhak, A., Krco, S., Theodoridis, E., and Pfisterer, D. Smartsantander: Iot experimentation over a smart city testbed. *Computer Networks 61* (2014), 217 – 238. Special issue on Future Internet Testbeds âĂŞ Part I.