What Can I Do Here?

IoT Service Discovery in Smart Cities

Edward Wang†
Electrical Engineering
University of Washington
Seattle, Washington 98195
Email: ejaywang@uw.edu

Richard Chow
Intel Corporation
Santa Clara, California
Email: richard.chow@intel.com

***Abstract*—The vision of smart cities and IoT is an environment blanketed with interconnected, software-enabled devices. Unlike software installed on personal devices, however, people may not know about services in the environment, or may not even be the intended users. People lack a unified way to discover software services in a smart city infrastructure, and the current device-centric approach to IoT is inconsistent with the growing network and software services associated with these devices. In this paper we outline changes needed in the current IoT framework to shift to a service model for IoT. We describe how, similar to users of a personal computing device, users can define their preferences, install services, and manage the data that is generated and consumed by services. In this framework, service preferences provide a basis for proper service discovery. As an illustration of the proposed model, we provide modifications to the well established Auto-ID Object Name Service and Physical Markup Language architecture to demonstrate how a practical system can support the concept of IoT services and discovery.**

I. Introduction

The vision of smart cities and the Internet of Things posits that the world will be blanketed with devices that collect information and interact with the world. These devices range from simple sensors to fully context-aware systems that provide end-to-end services. For end users, the model for software services with IoT is notably different from conventional computing. On a laptop or phone, the user installs software applications, and in principle can determine what applications are running and what data is collected. However, this model is reversed with the vision of IoT. People are no longer necessarily users of the IoT service, but instead may be the subject of the service, such as a smart city sound monitor. Another example is WiFi connectivity. The user discovers the availability of WiFi provided by companies such as Boingo, but are not the ones who install the actual access points. They subscribe to a service that another entity installs and maintains.

Extending that concept a bit further, consider an example of companies installing hardware to provide services such as occupant tracking via RFID or Bluetooth beacons. This could enable services such as family perimeter setting, to make sure children do not get lost in large public spaces like malls and parks. Similar to the Boingo example, the user might have subscribed to this service in the past, can determine which parks have this occupant tracking system, and can use it when available. In addition, the same mall might have futuristic cafes that subscribe to a sound and lighting control system that allows individual guests control over their environment.

Extending even further, in the vision for smart cities, millions of services exist, similar to the millions of applications that exist in app stores for phones today. With just a software install, new capabilities are unlocked for the phone. However, a major difference exists between the current device and app model and the IoT model: users choose what device will be installed and to a large extent the software services that use these devices. In a fully connected IoT world, however, this control over the devices and services available is lacking, especially in environments away from the user's home. This raises many design questions that involve the end user, such as usability, privacy, and discovery. How does the user learn about the available services in a public space, especially the ones that he actually wants to use? How will we streamline the user experience to go beyond downloading a separate app for the thousands of services a user cares about? To what level can the user still maintain privacy in hyperconnected environments where services are running with or without his or her consent?

To start, we need to change the way we conceptualize IoT, commonly thought of as a large number of devices that are connected to the Internet. Instead, we need to conceptualize IoT as a large number of services connected to Internet-enabled devices. In smart cities, the user is no longer involved in device or service installation. Hence, users must be able to discover services in the environment. In this paper we focus on enabling this discovery.

We describe a new paradigm, called the IoT Service Discovery Framework. To illustrate this service-oriented paradigm, we describe an architecture based on a modified version of Auto-ID Center's Object Name Service. We also describe a phone application that interacts with the IoT Service Discovery Framework to notify users of interesting nearby services. The following examples serve to motivate the need for our framework.

†Work done while at Intel Corporation

978-1-5090-1941-0/16/$31.00 ©2016 IEEE

A. Smart Environment Service Store

In the vision for IoT, everyday objects like chairs, cups, parking spaces, and in fact, literally everything in the environment becomes *smart* [1]. A smart chair can be morphed to fit the way you like. A smart coffee cup can adjust its temperature to fit your liking. Museum artifacts can be tagged with beacons that tell you more about them. By obtaining a user's preferences, a Service Store app can recommend new services that are interesting to the user. However, not everyone will want to be hyperconnected, nor are all services free.

We envision a Service Store app that allows users the ability to browse the services near the user. This app must filter these services based on the user's preferences: there are services the user always wants to use, services the user is interested in occasionally, and others that the user prefers not to use. In this way, these services can blend into the background, for example automatically morphing a chair to the user's liking moments before he sits down, because the system already knows the user always wants to use such services.

In order to communicate preferences and understand what services are available in the environment, the app uses our proposed lookup system.

B. Privacy Notification

One of the major privacy problems with IoT is the lack of awareness of when a device might be collecting personal data. The ubiquitous nature of IoT devices means that a person can easily not even know when sensors are present. A basic privacy tenet, dating back to the Fair Information Practice Principles [2], states that personal data collection should only happen with appropriate notice. One of the motivations of this paper is to enable the creation of a communication channel from IoT devices to people who are in the vicinity of a device that may be collecting personal data.

For instance, suppose a mall installs a suite of surveillance cameras in its parking structures for security purposes. Rather than describing the purpose of the cameras on a sign, the mall can register the cameras through the proposed lookup service. Passersby using a mobile app that interacts with the lookup service can be alerted to the presence of the surveillance cameras. This not only serves as a privacy notification, but can also provide a sense of security knowing that the area is safe due to the service.

Privacy notifications can also be incorporated into smart home systems. Consider open space voice controllers such as the Amazon Echo, which allows voice commands to play music and adjust lighting, for instance. Guests to the home may or may not be comfortable with a whole home voice recording system. Home owners can use the proposed lookup system to request a guest's permission to use such a service if their guests have indicated that they would like to be notified about audio recording in their vicinity. Depending on the guest's preference, the guest can be notified as they enter the house. If the guest indicates they are uncomfortable with the recording, the system can potentially disable the service temporarily or partially.

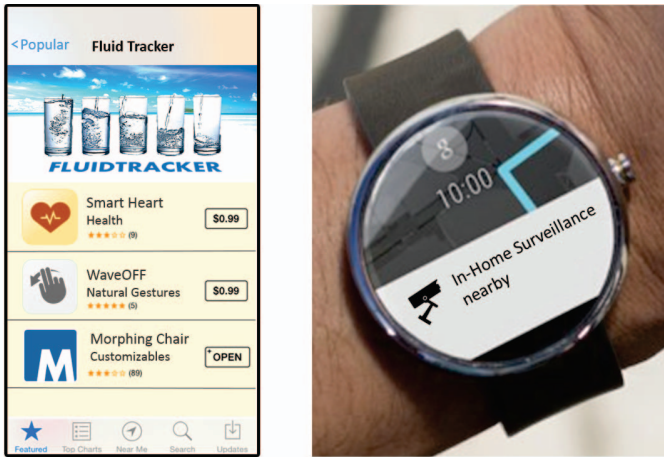

Fig. 1. Left: Service store use case for service discovery in the environment. A user can see relevant IoT services nearby. Right: User is notified on his watch of a nearby security camera when he enters his friend's home.

II. IoT Service Discovery Framework

Our motivating examples differ fundamentally from most current IoT services in one major way, the user does not install the service. Hence, the user does not have an understanding of what services are active in the environment. The focus of this framework is on the discovery of the existence of an IoT service near a user. Services that do not want to be discovered will not use this framework. Also, the framework does not focus on how these services work after they are discovered or what services use the data for.

We describe in this section architectural details of the IoT Service Discovery Framework. We describe how users can discover the services that are installed in the environment by someone else, and do not focus on how a service will interact with a user. In particular, we propose a set of modifications that can be made to the centralized lookup system developed by the Auto-ID Center, called EPCglobal [3].

A. Current EPCglobal architecture

Here we list the basic components of the EPCglobal architecture.

- Electronic Product Code (EPC) - Globally unique ID of a device [4].
- Product Markup Language (PML) - Markup language that describes information about a product using predefined tags [5]. Typical tags are dimensions, manufacturer, device classification, and control mechanisms.
- Object Name Service (ONS) - Name service that provides a lookup of the EPC and returns its corresponding PML [6]. This name service acts much like DNS, which resolves a domain name to an IP address. The result of the look up is a product description in the PML format.

B. Modifications to EPCglobal

To adapt EPCglobal infrastructure to our service-oriented framework, we propose the following changes.

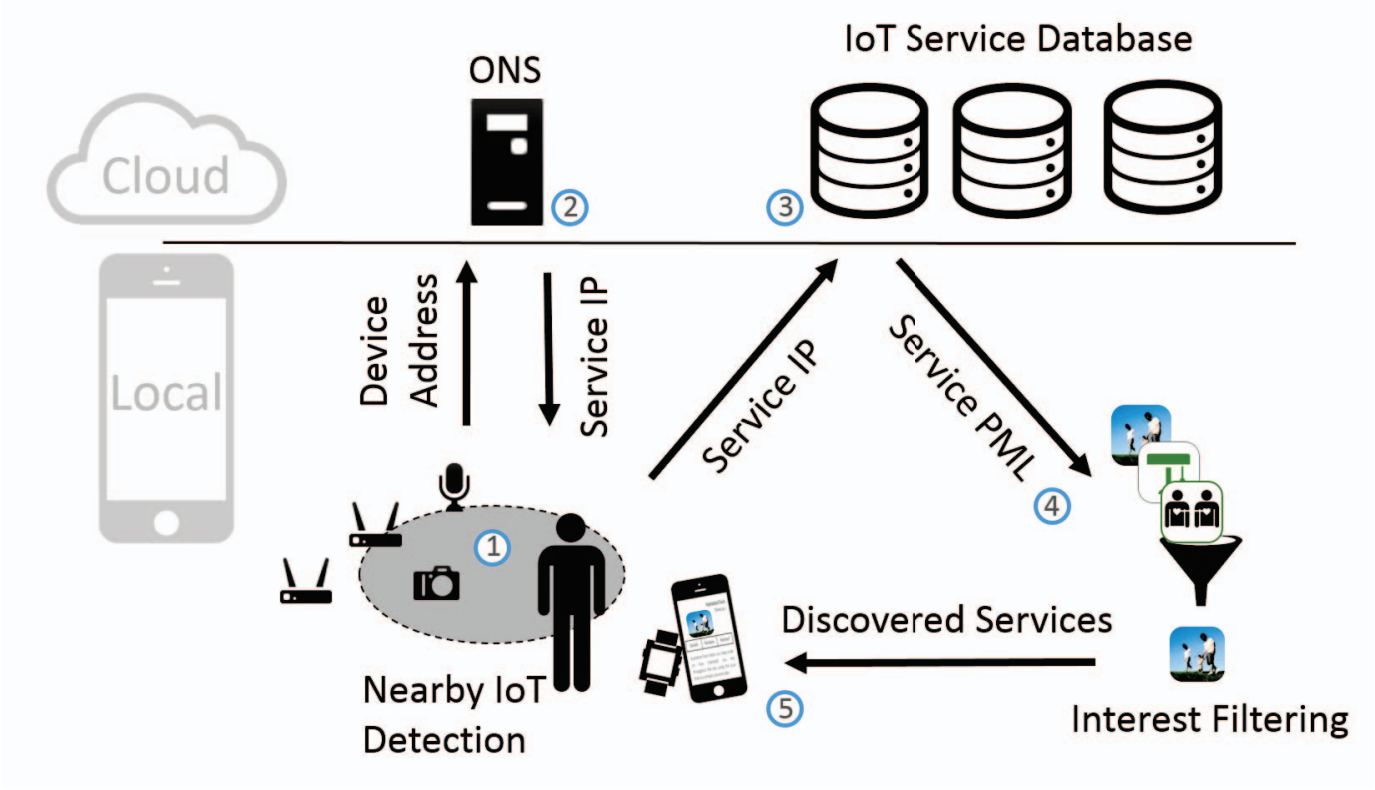

Fig. 2. (1) Nearby IoT devices are scanned for to retrieve a list of devices currently operating in the vicinity of the user. (2) The list of IoT devices are looked up in the Object Name Service (ONS). The ONS holds the mapping of the services the devices near the user are operating. From this lookup, a list of the IP address of the unique services are returned. (3) The service information are retrieved in the form of Physical Markup Language (PML). (4) The entire list of service PMLs are filtered based on preferences set by the user. (5) The filtered list of services are ultimately displayed to the user.

- The PML is a specification of a markup language to capture information about physical products. However, there have been additions to the original specifications over time as IoT has become more capable. For example, objects that afford actions have a control tag. In order to encapsulate services, we are proposing an additional tag for services. Service tags are also assigned EPC identifiers.
- Devices are mapped to a service by referencing the service's EPC in the device's PML.
- The service EPC can be looked up in ONS, returning the corresponding PML. This PML would contain the service metadata.

C. Service Tag

In the tag for services, we include service metadata, basic information about the service such as the service name, data used and generated, and service interaction hooks.

Our proposed service PML tag will follow the following format. The tag names below are for illustrative purposes only, but the metadata should contain the data collected and privacy policy.

```
<service epc = "x.xxx.xxx...">
  <name></name>
  <provider></provider>
  <description></description>
  <version></version>
  <device></device>
  <privacy></privacy>
  <data>
     <userdata></userdata>
     <raw></raw>
     <inference></inference>
     <type></type>
     <persistence></persistence>
  </data>
</service>
```

The EPC code is assigned to a service, and placed at the opening service tag. A service PML can then be referenced by this EPC:

```
<service epc = "x.xxx.xxx..." />
```

D. Device to Service Mapping

To retrieve the service EPC, note that we assume we can retrieve the device EPC. This retrieval is based on existing protocols for device discovery, for instance the RFID protocol assumed by EPCglobal.

In general, we support three major methods of device discovery:

1) **Beacons.** A beacon actively broadcasts the device's presence and provide a point of access to gain information about the device and how to interface with it. A commercial example is the iBeacon from Apple, which employs BLE to broadcast a device UUID that can be looked up by compatible applications for further information. The most prominent use case is promotion distribution at storefronts, but can be adapted as a technique for general device discovery.
2) **Tags.** A tag passively provides a device's information by being scanned. The barcode and QR codes are optically scanned while NFC and RFID tags are read by RF scanners.
3) **Gateways.** A gateway provides a portal for devices to access the Internet, using WiFi, Bluetooth, Zigbee, etc. In this way, the devices are only discoverable within a local network. Gateway can act as proxies for the devices running services. Hence, a gateway can register different services run by the devices connected to it under its own device ID. In this way, devices using gateways cannot be directly discovered but their services can.

The final output from device discovery is a list of device IDs of relevant devices around a person. This list of device IDs are looked up in the ONS for services being used by the set of devices.

Unlike the current ONS setup which only supports EPC, a more complete IoT service discovery architecture needs to support multiple ID conventions such as MAC Address, Bluetooth QPID, etc. This presents two issues. The first issue is that in our framework we require globally unique EPCs. If we want to incorporate multiple ID conventions, some processing must be done on the IDs to ensure global uniqueness. For instance, one could prepend the IDs with an ID convention identifier. The second issue is that some of these ID conventions are not necessarily globally unique. For instance, Bluetooth QPIDs are only statistically unique. One solution is for global uniqueness to be enforced at registration time.

E. IoT Service Discovery Architecture using EPCglobal

Figure 2 describes the overall flow of the architecture. Nearby IoT devices are retrieved as a list of device EPCs. The list of device EPCs are looked up in the ONS. The ONS returns the device's PML, which contains a list of service EPCs the device uses. Again using ONS, the service EPCs are looked up for their corresponding PML. From this lookup, all the PMLs of the services being used by a device will be retrieved. The entire list of service PMLs can then be filtered based on preferences set by the user. The filtered list of services are ultimately displayed to the user.

III. User and Service Interaction

Section II describes the foundation for a service discovery architecture. In this section, we describe aspects of user and service interaction that arise from having such an architecture in place.

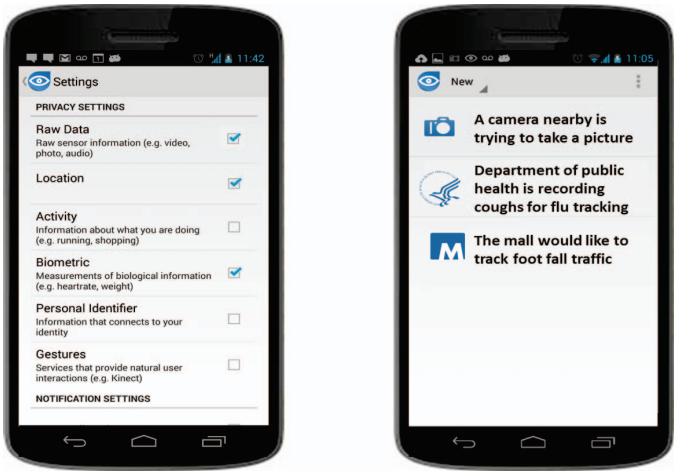

Fig. 3. Privacy Notification Application. Left: User can set preferences on the types of data collection that the user wants to be notified about. Right: Privacy notifications to the user.

We focus on two fundamental interactions: notification and enabling/disabling of services.

A. Notification

Using the framework, an administrator sets the service's visibility and declares the service metadata. The user interacts with the framework to retrieve the metadata. Privacy is a potential issue even if there is no intended user interaction (consider a security camera). Thus, a user may want a privacy notification depending on a service's metadata. The metadata may also signal to the user that the service may be of interest. In this way, a person can also be notified when they enter an environment with a service they are interested in, new or already subscribed.

B. Enable/Disable

An administrator sets whether users can interact with a service to enable or disable it, and whether the service is enabled or disabled by default. A surveillance system would be enabled by default with no capability for users to disable the service. In this case, users will only be notified about the service, but will have no ability to control it. In contrast, a family perimeter setting service might use the same surveillance system, but can be disabled by default, and enabled by a user through a payment.

Web browsers follow the HTTP protocol to retrieve a web page, but presentation of the fetched page depends on user and browser preferences. In the same way, an IoT service "browser" discovers IoT services and presents the discovered services to the user in some meaningful way. This can come in many forms. To illustrate this concept, we revisit the two motivating examples of an IoT service store and a privacy notification system. In each of these use cases, the user is assumed to have a mobile device, such as a smartphone, capable of discovering IoT devices through RF scanning, optical code reading, etc. We illustrate below how each of these service browsers can utilize the service discovery architecture.

The service store app on a smartphone scans the environment for new devices in its proximity. When it comes across a new device, it will perform an ONS look up of the device's ID, and retrieve a list of service information. Based on the user's preferences, a filter could be applied to only surface to the user the services of interest. The filter would look at the service PML fetched for each of the services. Preferences could be based on the software producer, such as for a customer loyal to the brand, or for functionality, such as services labeled for health tracking. If a user decides to then subscribe to a service, a service hook, such as a URL, can be fetched from the PML for enabling or paying for the service, or for downloading an app to control the service.

For an app responsible for privacy notification (see Figure 3), the app would retrieve service metadata in a manner similar to the service store app. The user would define in the privacy notification app's preferences what he or she would like to be notified about. This could include certain types of data being recorded such as cameras or audio or health information. From the service PMLs, information about the data recorded can be retrieved. If a service goes against the user's privacy preferences, the system can notify the user. The privacy policy of the service can be fetched from the PML and displayed to the user, as desired. For services that can be disabled, a service hook can be fetched from the PML that allows the user to communicate with the service. According to user preferences, the app might even be able to disable services automatically without notifying the user.

IV. Discussion

What is the relationship between the IoT Service Framework and device discovery?

The proposed service framework focuses on taking a set of devices around a user, and ultimately producing a list of services that surrounds the user. However, the framework is agnostic to the actual discovery mechanisms of the devices. These mechanisms are the subject of numerous IoT standards aiming at device interoperability and discovery [7], [8], [9].

Who declares the service metadata?

The service information declaration is usually the job of the service administrator. However, the manufacturer may provide metadata defaults to use in the declaration. For ease of use, these defaults may even be declared automatically by the device to the name service for home owners or small business owners, including the device to service mapping being registered on the ONS automatically when the service administrator installs the service. Note that declaration and registration of a service is optional, meaning a service administrator does not need to make the service discoverable.

How does the IoT Service Framework handle industry fragmentation?

The IoT ecosystem will likely remain fragmented with multiple competing vendors for any of the IoT Service Framework data flows described. This includes device discovery, which can come from different communication protocols. But the fragmentation can also come from the implementation of the device-to-service discovery mechanism, which in our described architecture uses the modified EPCglobal standard. In another possible implementation, devices might directly broadcast the service metadata, and there is no need to perform a name service lookup. This works well for devices that only use one service. Alternatively, another implementation could assign rotating IDs to a device to broadcast all the services the device uses. These IDs can then be looked up in the name service in turn. The architecture presented is simply one implementation for a service discovery system that can be adapted for systems that utilize EPC; there are multiple ways to implement each piece of our IoT Service Framework.

V. Related Work

Various industry efforts (e.g., IoT-A [8], BUTLER [10], AllJoyn [9], OIC [7]) are working on standardizing interoperability in IoT, allowing discovery and communication among devices. Apple's iBeacon [11] is another example of device discovery. A tagged device broadcasts an ID through Bluetooth and a compatible device can use this ID to retrieve associated information (e.g., for location-based services). Device discovery capabilities provided by iBeacon can also link a person with devices in the environment. These efforts are device-centric localization technologies. Our architecture incorporates and complements these efforts; our IoT service framework is agnostic to which local device detection technologies are used.

Our proof-of-concept architecture for discovery and information retrieval is based on EPCglobal, designed by the Auto-ID Center, which has over the last decade developed and refined an architecture for an RFID lookup system for Electronic Product Codes (EPC) called Object Name Service (ONS) [6], [12]. ONS acts much like DNS, but using EPC instead of hostnames, to ultimately serve information in the Physical Markup Language (PML) [5], [13] format. This includes information like manufacturer, dates, and even control of any actuation. This is naturally a one-to-one mapping because of the physical nature of the *objects*. Our work expands the use of ONS beyond physical devices to software services.

Ontological and semantic models that describe IoT devices and services have also been studied, for instance, see [14], [15]. Such semantic data models are complementary to our work and can be used to inform our PML-based service descriptions. See [16] for IoT service discovery using semantic models.

Various products in the IoT space intersect with components of our vision. Xively [17] proposes an open service framework, which enables users to tap seamlessly into various data feeds generated by their devices. EVRYTHNG [18] has an IoT cloud platform that tracks physical products and collects data from embedded sensors.

OpenIoT [19] proposes an IoT browser that can connect to a server platform, browse for IoT devices in any locality, and download related applications. Our architecture has some similar components, but we emphasize service discovery and

notification based on user preferences, as opposed to active browsing. OpenIoT, however, is also a good example of an implementation that fits into our IoT Service Framework.

Wirth et al. [20] have proposed an entirely local architecture for service discovery and interaction, interesting when Internet connectivity is only intermittent.

There is much work detailing the potential privacy concerns with IoT, for instance, see [21], [22]. Langheinrich [23] also describes a system with privacy notifications through a mobile privacy app and standardized privacy policies. There are some differences in architecture (e.g., privacy proxy in the cloud rather than on the mobile device), but in general the privacy goals of Langheinrich's system are similar to ours. The main difference is that we propose to integrate privacy communication with service discovery and interest filtering, perhaps a more practical approach compared to using dedicated systems.

VI. Conclusion and Future Work

We have proposed a unified service discovery architecture for IoT services. This will become more important as users find themselves in smart environments beyond the home. We described abstractly the properties and requirements of an IoT Service Discovery Framework. We then provided details on a particular implementation, which is a refinement of an existing lookup architecture developed by Auto-ID for RFID Electronic Product Code lookup for physical devices. IoT applications and services can span multiple physical devices, and a single physical device can run multiple services. Hence, the Auto-ID solution or any device-centric architecture is not suited for IoT. As part of our proposed architecture, we incorporated existing service discovery protocols, which are fragmented based on various proximity and local communication technologies.

As an important application of our service discovery architecture, we proposed a system with the ability to provide notice-and-consent for IoT. A basic privacy principle is that personal data collection should only happen with appropriate notice. However, the implementation of this principle, already difficult in general, is even more difficult for IoT as there is no natural communication channel with users. In many cases, users may be in an environment where services are running and capturing their data even though the users may not have installed the services and may not even know they are running.

We described a mobile application that allows end users to interact with our service discovery infrastructure. The application allows automatic discovery of IoT services in the user's environment. Depending on settable preferences, the application alerts the user when services interesting to the user are in the vicinity, or if there is data collection happening that is potentially troublesome to the user.

This paper is only a step towards a general framework for IoT service discovery. Many challenges remain. For instance, more work is needed to investigate two key aspects of our proposed system: system performance and the user interface. Latency and bandwidth needs to be practical and scalable for service lookups, given the large number of devices and services in a smart city. Also deserving study is the data schema and how to represent the plethora of IoT services in an extendable and meaningful way. This is critical both for effective interest filtering and for privacy.

References

[1] G. Kortuem, F. Kawsar, V. Sundramoorthy, and D. Fitton, "Smart objects as building blocks for the internet of things," *IEEE Internet Computing*, vol. 14, no. 1, pp. 44–51, Jan. 2010. [Online]. Available: http://dx.doi.org/10.1109/MIC.2009.143

[2] "Fair information practice principles," https://en.wikipedia.org/wiki/FTC_Fair_Information_Practice.

[3] J. Shi, Y. Li, and R. H. Deng, "A secure and efficient discovery service system in epcglobal network," *Comput. Secur.*, vol. 31, no. 8, pp. 870–885, Nov. 2012. [Online]. Available: http://dx.doi.org/10.1016/j.cose.2012.08.005

[4] GS1, "Electronic product code," http://www.gs1.org/sites/default/files/docs/epc/TDS_1_9_Standard.pdf, 2014.

[5] D. L. Brock, "The physical markup language," http://xml.coverpages.org/PML-MIT-AUTOID-WH-003.pdf, 2001.

[6] GS1, "Object name service," http://www.gs1.org/sites/default/files/docs/epc/ons_2_0_1-standard-20130131.pdf, 2013.

[7] "Open interconnect consortium," http://openinterconnect.org/.

[8] "Iot-a," http://www.iot-a.eu/public.

[9] "Allseen alliance," https://allseenalliance.org/.

[10] "Butler," http://www.iot-butler.eu/.

[11] "iBeacon," https://developer.apple.com/ibeacon/.

[12] B. Fabian, "Secure name services for the internet of things," Ph.D. dissertation, Humboldt University of Berlin, 2008. [Online]. Available: http://edoc.hu-berlin.de/dissertationen/fabian-benjamin-2008-08-07/PDF/fabian.pdf

[13] D. Brock, T. Milne, Y. Kang, and B. Lewis, "The physical markup language core components: Time and place," Auto-ID Center, MIT, Cambridge, USA, Technical Memo, 2001.

[14] S. De, P. M. Barnaghi, M. Bauer, and S. Meissner, "Service modelling for internet of things," in *Federated Conference on Computer Science and Information Systems - FedCSIS 2011, Szczecin, Poland, 18-21 September 2011, Proceedings*, 2011, pp. 949–955. [Online]. Available: http://fedcsis.eucip.pl/proceedings/pliks/113.pdf

[15] S. Mayer and D. Guinard, "An extensible discovery service for smart things," in *Proceedings of the Second International Workshop on Web of Things*, ser. WoT '11. New York, NY, USA: ACM, 2011, pp. 7:1–7:6. [Online]. Available: http://doi.acm.org/10.1145/1993966.1993976

[16] A. Zaslavsky and P. P. Jayaraman, "Discovery in the internet of things: The internet of things (ubiquity symposium)," *Ubiquity*, vol. 2015, no. October, pp. 2:1–2:10, Oct. 2015. [Online]. Available: http://doi.acm.org/10.1145/2822529

[17] "Xively," http://xively.com/.

[18] "Evrythng," https://evrythng.com/.

[19] J. Kim and J.-W. Lee, "OpenIoT: An open service framework for the internet of things," in *Internet of Things (WF-IoT), 2014 IEEE World Forum on*, March 2014, pp. 89–93.

[20] H. Wirtz, J. Rüth, M. Serror, J. A. Bitsch Link, and K. Wehrle, "Opportunistic interaction in the challenged internet of things," in *Proceedings of the 9th ACM MobiCom Workshop on Challenged Networks*, ser. CHANTS '14. New York, NY, USA: ACM, 2014, pp. 7–12. [Online]. Available: http://doi.acm.org/10.1145/2645672.2645679

[21] J. H. Ziegeldorf, O. G. Morchon, and K. Wehrle, "Privacy in the internet of things: threats andchallenges," *Security and Communication Networks*, vol. 7, no. 12, pp. 2728–2742, 2014. [Online]. Available: http://dx.doi.org/10.1002/sec.795

[22] M. Henze, L. Hermerschmidt, D. Kerpen, R. Huling, B. Rumpe, and K. Wehrle, "A comprehensive approach to privacy in the cloud-based internet of things," *Future Generation Computer Systems*, pp. –, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X15002964

[23] M. Langheinrich, "A privacy awareness system for ubiquitous computing environments," in *Proceedings of the 4th International Conference on Ubiquitous Computing*, ser. UbiComp '02. London, UK, UK: Springer-Verlag, 2002, pp. 237–245. [Online]. Available: http://dl.acm.org/citation.cfm?id=647988.741491