Web-Based Management of the Internet of Things

Lina Yao and Quan Z. Sheng • *University of Adelaide*

Schahram Dustdar • *Vienna University of Technology*

The Internet of Things (IoT) system presented here seamlessly integrates virtual and physical worlds to efficiently manage things of interest (TOIs), where services and resources offered by things easily can be monitored, visualized, and aggregated for value-added services by users. Using practical experience gained from this system, the authors identify several R&D opportunities for building future IoT applications.

As noted by Tim Berners-Lee, the inventor of the World Wide Web, it's not the documents that are actually interesting, it's the things they're about.[1] The last level of abstraction for the Web is to connect physical things. With recent advances in RFID, wireless sensors networks, and Web services, the Internet of Things (IoT) offers the capability of integrating information from both the physical and virtual worlds. With IoT, it becomes possible to infer the status of real-world entities with minimal delay using a standard Web browser.[2,3]

IoT is the embodiment of the evolution from systems linking digital documents to systems relating digital information to real-world physical items. While it's well understood that IoT offers numerous opportunities and benefits, it also presents significant technical challenges in developing applications in the IoT environment (see the related sidebar discussing others' efforts).[2,4] One of the main challenges is to smoothly and seamlessly integrate the virtual and physical worlds to effectively manage things of interest (TOIs) in IoT.[5-7] This is critical for a number of important applications, such as object discovery (for example, finding a quiet restaurant), recommendation (suggesting a device that can consume a video stream), and mashup (composing device functionalities for a new service). A crucial prerequisite is acknowledging the seamless information access, exchange, and manipulation between the two worlds.

Here, we explore a new direction in realizing information integration between the physical and virtual worlds. We design and develop a system that offers an integrated Web-based interface to manage (that is, connect, monitor, control, mashup, and visualize) things in an IoT environment, which helps people be aware of their surroundings and thereby make better decisions. The system provides a layered framework for managing and sharing the information produced by physical things. We also adopt a rule-based approach to aggregate individual things for building context-aware, personalized new value-added services. We implement the prototype in a real-world, inhabited home environment, where residents can access, control, and compose physical resources in a variety of contexts. The practical experiences gained from this IoT project provide insight into how applicable IoT is to real-world industrial applications such as independent living of the elderly, healthcare, and environmental monitoring of smart cities.

Our System Design

Figure 1 shows our system's architecture. The architecture is layered, developed using the Microsoft .NET framework and SQL Server 2012. The system maps physical things and their related data to corresponding virtual resources, which it can aggregate and visualize via a range of software components. The system provides two ways

Related Work in Developing IoT Applications

With billions of things interconnected and present over the Web, there are significant software engineering challenges in developing Internet of Things (IoT) applications, due to their unique and inherent characteristics. The SENSEI project (www.sensei-project.eu) proposes an architectural framework that focuses on addressing scalability issues in wireless sensor and actuator networks. SemSorGrid4Env (www.semsorgrid4env.eu) develops a service-oriented architecture and middleware that assists developers in building large-scale, semantic-based sensor network applications. Both projects, however, deal with the connectivity issues of IoT: how to connect heterogeneous things to the Web rather than how to describe and model things. The recent research and development activities at the *Commonwealth Scientific and Industrial Research Organization (*CSIRO)[1] offer some interesting experiences in applying IoT in a number of application domains, such as smart farming. They developed an ontology-enabled architecture where the sensor observations are published as linked data cubes for long-term data analysis and sharing at the national scale. However, the system doesn't provide suitable integrated abstractions for things.

University of Washington researchers developed an IoT application, which unfortunately only focuses on managing the collected RFID data.[2] The Paraimpu platform[3] provides a Web-based social platform for people to connect, compose, and share things. To perform mashups of heterogeneous things, Paraimpu exploits several strong abstractions. In the Hyperpipe project (http://geoweb.crs4.it/doku.php?id=hyperpipes), things are represented as Web services and connected using pipes so that users can easily compose things for new services. However, most things are resource-constrained and the traditional service-oriented architecture (SOA) standards like SOAP and Business Process Execution Language (BPEL) might not be applicable. Many research projects are actively solving these challenges and one notable effort is the IoT6 project (www.iot6.eu), which focuses on investigating IPv6 and related standards, such as the Constraint Application Protocol (CoAP)[4] and IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN; http://tools.ietf.org/wg/6lowpan) to overcome IoT's current fragmentation.

References

1. K. Taylor et al., "Farming the Web of Things," *IEEE Intelligent Systems*, vol. 28, no. 6, 2013, pp. 12–19.
2. E. Welbourne et al., "Building the Internet of Things Using RFID: The RFID Ecosystem Experience," *IEEE Internet Computing*, vol. 13, no. 3, 2009, pp. 48–55.
3. A. Pintus, D. Carboni, and A. Piras, "Paraimpu: A Platform for a Social Web of Things," *Proc. 21st Int'l World Wide Web Conf.*, 2012, pp. 101–104.
4. C. Bormann, A.P. Castellani, and Z. Shelby, "CoAP: An Application Protocol for Billions of Tiny Internet Nodes," *IEEE Internet Computing*, vol. 16, no. 2, 2012, pp. 62–67.

to identify physical objects and connect them to the Web. The first is to use RFID technology, where physical objects are attached with RFID tags and interrogated by RFID readers. The second is to combine sensors with objects to transfer the raw data to the network. Then the raw data captured by readers and sensors is processed further. In the following, we describe the key system modules and their implementation details.

Data Access and the Sensor Hive

The data access layer manages RFID tags and sensors associated with physical things; collects raw RFID and sensor data and processes them; and provides a universal API for higher-level programs to retrieve the status of things. Due to inherent characteristics of RFID and sensor data (it's volatile, for example),[8,3] this layer contains several software components for filtering and cleaning the collected raw data, and adapting such data for high-level applications. The advantage of the data access layer is to allow the system to provide data synchronously. This is important, since some devices work with more than one sensor and the sensor readings may come asynchronously. This layer works in a scalable, plug-and-play fashion, where we can easily plug in new sensors and remove the old sensors.

The sensor hive module in the architecture essentially lets you map physical things to the corresponding virtual resources. To achieve a seamless integration of physical things with an organization's business processes, applications must be able to understand different data semantics. There are several languages, such as Physical Markup Language (http:// web.mit.edu/mecheng/pml), Microformats (www.microformats.org), and Resource Description Framework (RDF; www.w3.org/RDF), which we can use to add semantics to the descriptions of things. In our design, we exploit schema.org (www.schema.org), a recent initiative (launched in 2011 by Bing, Google, and Yahoo) that's designed for both human users and machines. The universal RESTful API provided in our design enables higher-level programs to retrieve the status of physical things with specified addresses, without knowing where and how to find the physical sensors associated with the things.

Virtual Things

This module maps a collection of classes (also called *virtual things*) to their corresponding physical things. Each virtual thing communicates with
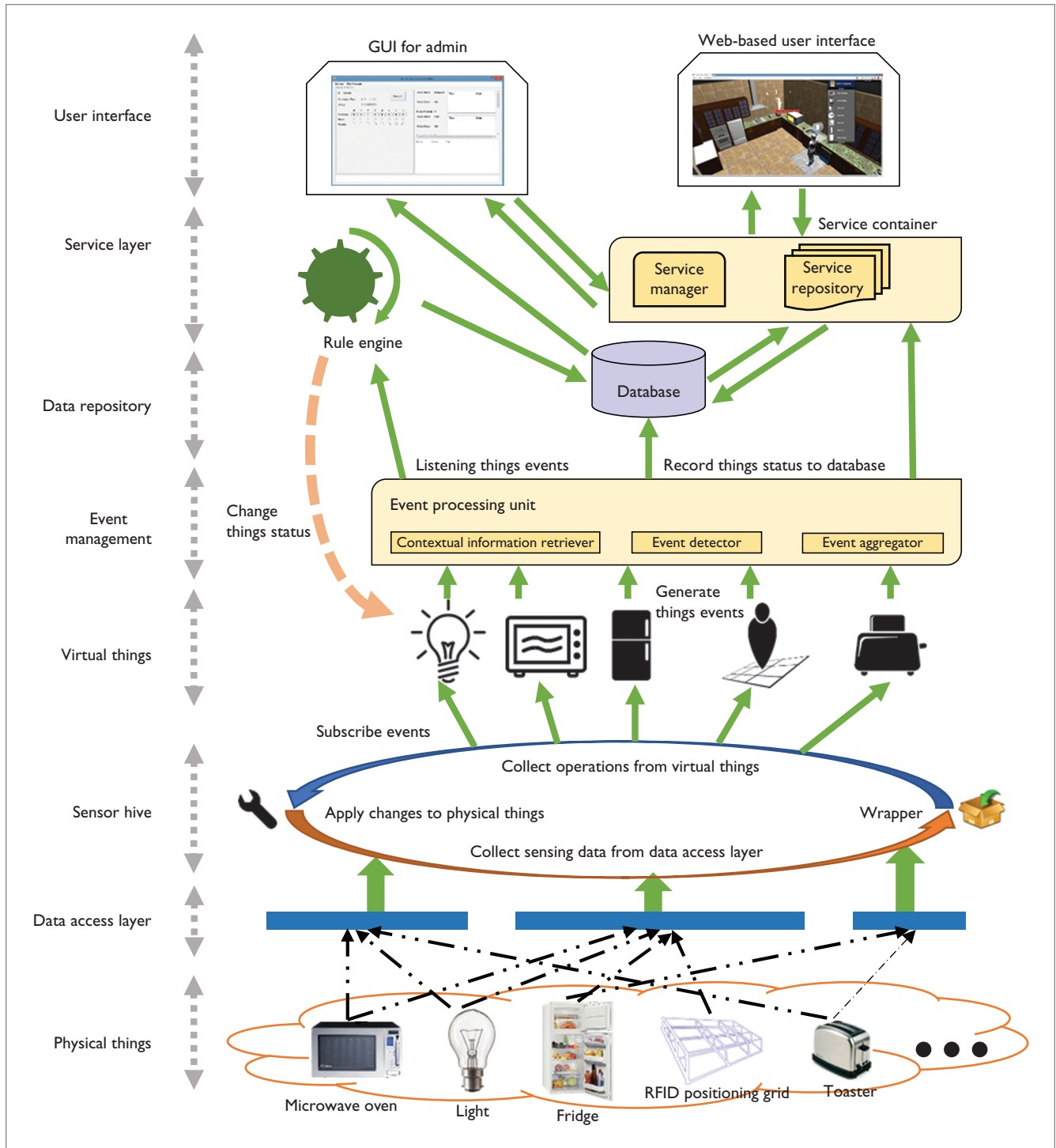
Figure 1. Our system's layered architecture. The system maps physical things and their related data to corresponding virtual resources, which it can aggregate and visualize via a range of software components.

the sensor hive, collects the information, and interprets the current status of the corresponding physical device. For example, the virtual device of a microwave oven can query the sensor values associated with the physical microwave oven from the sensor hive layer, and use this information to decide the oven's current status (idle or in use). A virtual thing is responsible for generating events based on the collected information (such as when the refrigerator door opens), which can be directly used by high-level applications or further

processed by other modules (for example, an event-processing unit).

Event Management

This layer focuses on event processing that automatically extracts and aggregates objects' usage events based on the data feeds from the virtual things layer in a pipelined fashion. The pipeline consists of three main phases: *event detection*, *contextual information retrieval*, and *event aggregation*.

The event detector captures and decides whether a physical thing is in use. In our design, there are two ways to detect usage events of things: a *sensor-based approach* for detecting state changes and *RFID-based approach* for detecting mobility.[7] In sensor-based detection, an object's usage is reflected by changes of the object's status — for example, the status of a microwave oven moves from ideal to busy when it's being used. In the RFID-based detection, an object's movement indicates that the object is being used. For example, if a coffee mug is moving, it's likely that the mug is being used. In this situation, we adopt a generic method based on comparing descriptive statistics of the Received Signal Strength Indication (RSSI) values in consecutive sliding windows.[9] The statistics obtained from two consecutive windows are expected to differ significantly when an object is moved.

The contextual information retriever extracts contextual information contained in things' usage events. In our current design, we focus on three types of contextual information: *identity* (user), *temporality* (time stamp) and *spatiality* (location).[9] To obtain the identity information, we perform manual labeling, where all participants mark and record their activities. To obtain the temporal information, we split a day into 48 equal intervals of half an hour each. For example, if the time stamp of a usage event is 9:07 am,

it will be assigned to the 9:00–9:30 am interval. For the spatial information, we consider two situations. For *static* objects (perhaps a refrigerator or microwave oven), the spatial information is a prior knowledge. For *mobile* objects (an RFID-tagged coffee mug, for example), we provide coarse-grain and fine-grain methods for localization. The coarse-grain method uses the RSSI signal received from a tagged object to approximate its proximity to an RFID antenna. Each zone is covered by a mutually exclusive set of RFID antennas. The zone scanned by an antenna with the maximum RSSI signal is regarded as the object's location. The fine-grain method compares the signal descriptors from an object at an unknown location to a previously constructed radio map or fingerprint. We use the weighted k Nearest Neighbors algorithm (w-kNN) to find the most similar fingerprints and compute a weighted average of their 2D positions to estimate the unknown tag location.[9]

The event aggregator indexes and stores all the events and services, together with their related information in a database (in the data repository layer), which can be mined for various purposes (for example, finding *latent correlations* among things, and making recommendations).[9] A list of elements is constructed, storing the identifiers of objects, their types and values, as well as the calculated contextual information. In this way, applications can focus on the functionalities without worrying about operations, such as connecting to the database, opening connections, querying with specified languages, and handling the results (normally they're raw data and inconvenient to access).

The Service Layer

This layer consists of a *rule engine* and a *service container*. The service container converts events and data into corresponding services. In particular,

the repository stores the descriptions of services (in the form of RESTful APIs) for things, where applications can easily access the data associated with a particular thing stored in the database (for example, usage history of a device), and manipulate the actuators (such as turning on or off a light). The APIs are represented using JavaScript Object Notation (JSON), which is developed from JavaScript for representing simple data structures and associative objects. The service manager is responsible for abstracting services from the lower level, representing them as services, and storing them into the service repository.

The rule engine allows applications to control a device automatically by establishing a set of rules. A rule consists of two parts: a *condition* and an *action*. A condition is a composition of a set of simple boolean expressions. An action is simply a set of device settings — for example, `Out-DoorLight.On=true` will turn on the outdoor lighting device. Alternatively, an action can also be a program — for example, `SendEmail(Microwave, 'someone@example.com')` will send an email to the email address. By combining simple Boolean expressions together, the application can setup a complex rule to make devices "smarter."

The rule engine consists of three main components: the *rule composer*, the *rule interpreter*, and the *rule parser*. The rule composer is a Web-based application implementing a user-friendly GUI for rule creation and action setup, in a drag-and-drop fashion. The rule interpreter is software that receives the string expressions of rules from the rule composer. It analyzes and annotates the string statement based on a state machine. The string expression is then translated to a list of annotated objects. The rule parser is implemented based on the shunting-yard algorithm. It first compiles each part of the input sequence into a .NET Expression
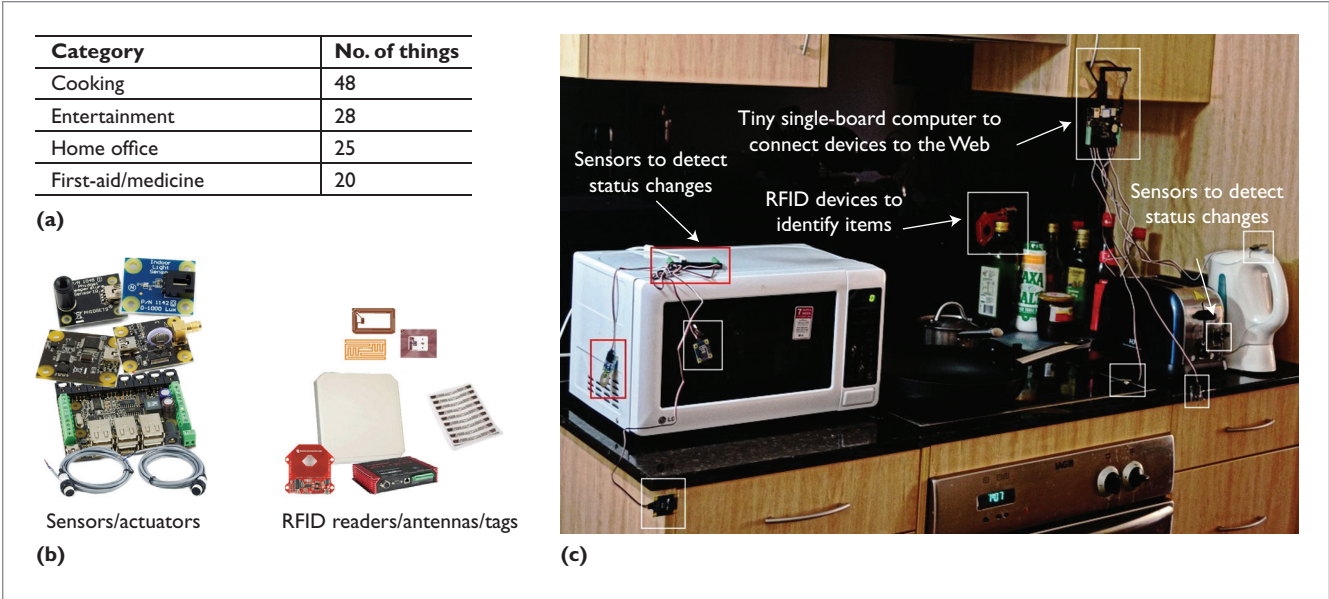
Category	No. of things
Cooking	48
Entertainment	28
Home office	25
First-aid/medicine	20

(a)

Sensors/actuators

RFID readers/antennas/tags

(b)

Tiny single-board computer to connect devices to the Web

Sensors to detect status changes

RFID devices to identify items

Sensors to detect status changes

(c)

Figure 2. The settings of the smart home Internet of Things (IoT) application. (a) Statistics of the involved physical things, (b) some devices used in the application, and (c) part of the kitchen setting, such as an IoT-enabled microwave oven and toaster.

object. Then, it combines all such objects together into a complex Expression Tree, which will be compiled into a Lambda expression. This Lambda expression object will be stored in memory when the system is running. It can be invoked when a device status changes or time elapses. If the Lambda expression returns true, a corresponding action will be called.

The User Interface Layer

This layer provides access to the management of things (for example, connection, monitoring, control, mashup, and visualization). The Web-based interface offers a 3D scene in a Web browser for users to manage their things of interest. We particularly adopt the Web Graphics Library (WebGL) in HTML5 to enable 3D scene recreation. The 3D models are stored as Digital Asset Exchange (DAE) files, and imported and rendered by using three.js (http://threejs. org) with plugins. Things are visualized and managed by device plugins. Each visualized thing is considered as a device plugin, which contains one or more 3D model or animation

settings. For instance, the kettle will show steam when it's boiling water. We can use the `ShaderParticle-Engine` plugin (https://github.com/ squarefeet/ShaderParticleEngine) for three.js to create the steam effect for the kettle. Each device plugin also provides a serial of APIs, to communicate with the service layer for status changes of the corresponding things, and to reflect such changes on the Web browser. This layer also provides an administrative interface for things management (for example, connecting and disconnecting things, and viewing event logs).

A Demonstrator IoT Application

We developed a number of IoT applications based on the proposed system architecture. Here, we particularly focus on introducing a *smart home* application, which has been successfully deployed in a real-world environment (at the first author's home). We attached RFID tags and sensors to 121 physical things (a microwave oven, fridge, coffee mug, laptop, couch, and so on) and Figure 2a shows

the statistics of the involved things. Figure 2b shows some RFID devices and sensors used in the implementation, and Figure 2c shows part of the kitchen setting, including an IoT-enabled microwave oven and toaster. When users interact with these physical things, events are captured (by RFID readers, for example). The raw data of the events must be processed (through cleaning, transformation, and integration) before storing in the repository. This task greatly benefits from our extensive experience in a recently completed, seven-year RFID research project.[10]

This application offers an integrated Web-based interface where a user can monitor and visualize the status changes of household things of interest in real time. Users can monitor, track, and control the current status of physical things by directly observing the status of their corresponding icons from the Web browser. In addition, this application also augments the physical things with key social network functionality. We developed a real-time notification of things' status by exploiting

the Twitter API. Status changes can be sent to Twitter subscribers in real time. Figure 3a shows the integrated Web-based user interface.

Real-Time Visualizer and Monitor

This function offers access to, and control of, the physical things, allowing the status of physical things to be visualized in real time. We render the icons with different effects to be consistent with the real status of physical things. For example, if the microwave oven is being used, its corresponding icon in the Web interface will change to a highlighted status (yellow), and otherwise gray (see part 1 of Figure 3a).

The learned position information from the event-processing module (see Figure 1) is used to visualize the locations and traces of TOIs at a coarse-grain level. We divide the testing area into multiple grids in a chessboard style (for example, 3 m * 3 m in the kitchen area). Then, the trace line is generated by connecting the grids' central dots. For example, as part 2 of Figure 3a shows, the trace of a coffee mug held by a subject is displayed in green lines.

Rules Composer

This function provides a graphical interface letting users control the devices by setting up a series of rules via a Web browser without any programming efforts (see part 3 in Figure 3a) with the details shown in Figure 3b. It can create a composite service using the rule-based composition component provided by the system, which consists of a widget panel (on the left) and a rules editor panel (on the right). The widget panel shows the virtualized objects (things and people) and each object has a set of actions (the light, for example, is associated with two actions: turn on and turn off). Users simply drag any object's widget to the rules editor panel and start to create a new rule or change the old rules by clicking the "edit" button beside each rule.
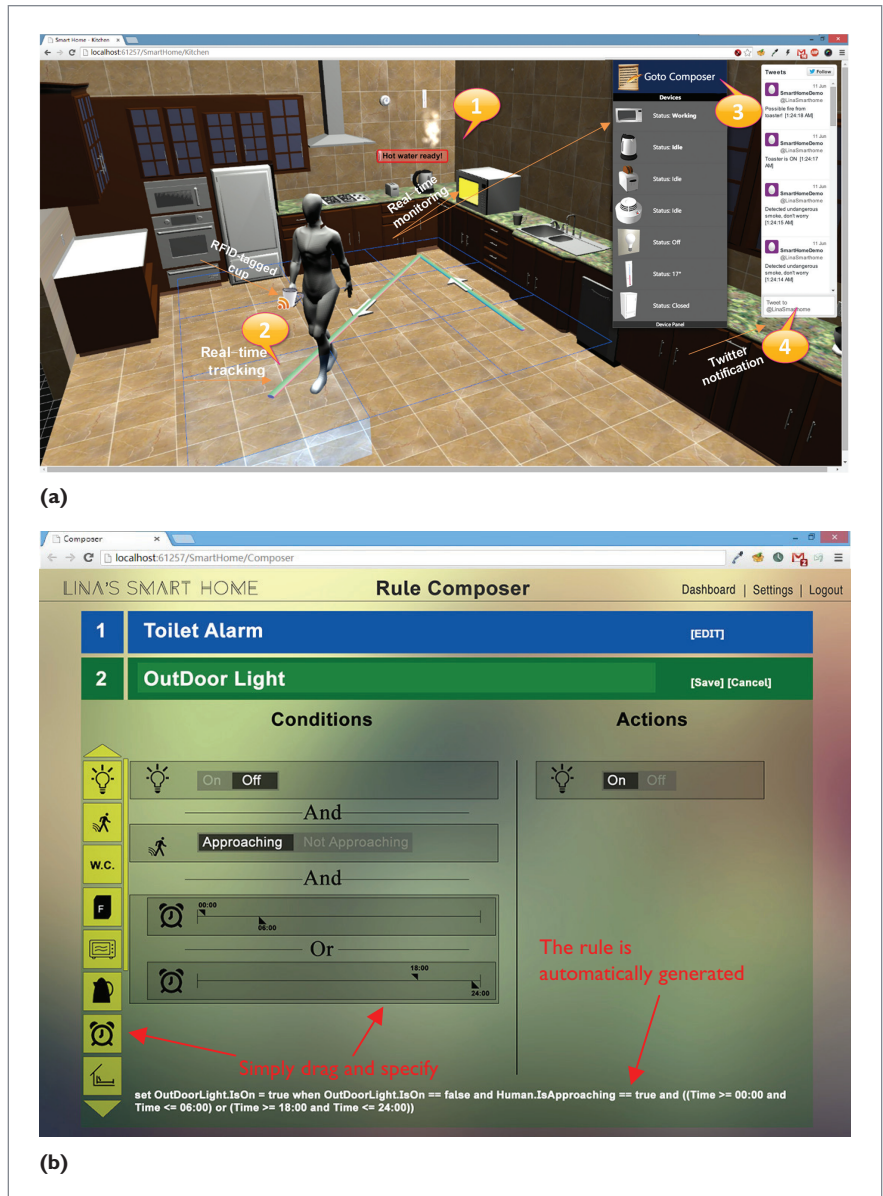

(a)

(b)

Figure 3. Real-time notification of things' status using Twitter. (a) The 3D Web-based application interface. (b) The rule composer supports users so that they can easily create their own rules. For example, for editing a rule like "turn on the outdoor light when a person is approaching during midnight to 6 am or 6 pm to midnight," a user needs only to drag the person and clock icons to the condition subpanel and the light icon to the action subpanel, and perform some simple adjustments (such as adjusting the clock slider to set the time period).

An editing panel includes a condition editor and an action editor. The condition editor lets users change or set rules based on contextual information (for instance, environmental information such as temperature, or event information such as a person approaching). Figure 3b shows the interface on specifying a rule.

Over the last few years, IoT has become a vibrant and rapidly expanding area of research and

development. We view the practical experiences gained, and the lessons learned, from our work presented in this article as a step towards further effective development of IoT applications. IoT is still far from mature, and building IoT software calls for more innovative solutions that address many remaining challenges. Here, we identify several directions for future research and development.

Things Discovery in IoT. Physical things are becoming an integral part of the emerging ubiquitous Web. With billions of things connecting and interacting over the Internet, there's an urgent need to effectively index, organize, and manage them for discovery, recommendation, and mashups. A fundamental task is to discover underlying connections among things, which remains a significant challenge. Indeed, finding things' correlations is a much more challenging task than finding relations for Web documents, for two reasons. First, things are diverse and heterogeneous in terms of functionality, access methods, and descriptions. Despite recent efforts in semantic techniques, the profiles of things still can't be crafted and represented easily in a meaningful feature space.[5] Second, correlations among things aren't obvious and are difficult to discover. Unlike social networks of people, where users have observable links and connections, things often exist in isolated settings and the explicit interconnections between them are typically limited. Based on our experience, we further investigated this challenge by proposing a novel graph-based method to mine the rich content embodied in human-thing interactions. Some preliminary results from our work are reported elsewhere.[7,9,11]

Data Management in IoT. Managing IoT data is another fundamental challenge in building IoT software. IoT data is not only extremely large in scale and volume, but also continuous, distributed, streaming, and volatile. Indeed, IoT is one of the major contributors to the age of Big Data. Given the scale of IoT, data management topics such as storage, real-time data stream analytics, event processing, data quality and uncertainty, and data interpretation and aggregation all need a revisit.[2] We recently conducted a survey[4] to investigate the main techniques and the state-of-the-art research efforts in IoT, particularly from data-centric perspectives, including data stream processing, data storage models, complex event processing, and searching in IoT. Several open research issues on IoT data management are also discussed.

Transaction Handling in IoT. In an IoT environment, the physical and virtual worlds co-exist and interact simultaneously. To process and manipulate information seamlessly between the two worlds, large amounts of data must flow between both worlds to ensure they're synchronized. This calls for new challenges to process heterogeneous data streams to materialize real-world events in the virtual world, and to send interesting events from the virtual world to users in the physical world.[12] In addition, most things are resource-constrained, which are typically connected to the Internet using lightweight, *stateless* protocols such as the Constraint Application Protocol (CoAP)[13] and IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN; http://tools.ietf.org/wg/6lowpan), and accessed using RESTful Web services. This makes transaction handling a challenging task, and extensive research is needed in this direction.

Security and Privacy in IoT. Due to the proliferation of smart devices in IoT, security and privacy protection is a serious challenge in building IoT applications.[2,4,8] In IoT, everyday objects become information security risks. In addition, more and more personal devices have embedded sensing capabilities for people's activities, locations, and surrounding environments. Recent advances in cryptographic algorithms can make general-purpose processors encrypt packets at line rates. How to exploit such algorithms in IoT still remains a challenge, since things in IoT normally possess low transmission rates and their connections are lossy. We believe that comprehensive solutions are needed not only to protect IoT information and give users the confidence that their data won't be misappropriated, but also to maintain desirable system performance. ⬚

References

1. T. Berners-Lee, "The Web of Things," *ERCIM News*, no. 72, 2008; http://ercim-news.ercim.eu/en72/keynote/the-web-of-things.
2. L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A Survey," *Computer Networks*, vol. 54, no. 15, 2010, pp. 2787–2805.
3. P. Barnaghi, A. Sheth, and C. Henson, "From Data to Actionable Knowledge: Big Data Challenges in the Web of Things," *IEEE Intelligent Systems*, vol. 28, no. 6, 2013, pp. 6–11.
4. Y. Qin et al., "When Things Matter: A Data-Centric View of the Internet of Things," *CoRR*, July 2014; http://arxiv.org/abs/1407.2704.
5. B. Christophe, V. Verdot, and V. Toubiana, "Searching the 'Web of Things'," *Proc. 5th Int'l Conf. Semantic Computing* (ICSC), 2011, pp. 308–315.
6. A. Pintus, D. Carboni, and A. Piras, "Para-impu: A Platform for a Social Web of Things," *Proc. 21st Int'l World Wide Web Conf.*, 2012, pp. 101–104.
7. L. Yao and Q.Z. Sheng, "Exploiting Latent Relevance for Relational Learning of

Ubiquitous Things," *Proc. 21st ACM Int'l Conf. Information and Knowledge Management*, 2012.

8. Q.Z. Sheng, X. Li, S. Zeadally, "Enabling Next-Generation RFID Applications: Solutions and Challenges," *Computer*, vol. 41, no. 9, 2008, pp. 21–28.

9. L. Yao et al., "A Model for Discovering Correlations of Ubiquitous Things," *Proc. 13th IEEE Int'l Conf. Data Mining*, 2013, pp. 1253–1258.

10. Y. Wu et al., "Modeling Object Flows from Distributed and Federated RFID Data Streams for Efficient Tracking and Tracing," *IEEE Trans. Parallel and Distributed Systems* (TPDS), vol. 24, no. 10, 2013, pp. 2036–2045.

11. L. Yao et al., "Exploring Recommendations in Internet of Things," *Proc. 37th ACM SIGIR Conf.*, 2014, pp. 855–858.

12. B.C. Ooi, K.-L. Tan, and A.K.H. Tung, "Sense the Physical, Walkthrough the Virtual, Manage the Co(existing) Spaces: A Database Perspective," *SIGMOD Record*, vol. 38, no. 3, 2009, pp. 5–10.

13. C. Bormann, A.P. Castellani, and Z. Shelby, "CoAP: An Application Protocol for Billions of Tiny Internet Nodes," *IEEE Internet Computing*, vol. 16, no. 2, 2012, pp. 62–67.

Lina Yao is a lecturer in the School of Computer Science at the University of Adelaide. Her research interests include data mining, the Internet of Things, ubiquitous computing, and service-oriented computing. Yao has a PhD in computer science from the University of Adelaide. Contact her at lina.yao@adelaide.edu.au.

Quan Z. Sheng is an associate professor and head of the Advanced Web Technologies Research Group in the School of Computer Science at the University of Adelaide. His research interests include the Web of Things, Big Data analytics, Web science, service-oriented computing, pervasive computing, and sensor networks. Sheng has a PhD in computer science from the University of New South Wales. He's a member of IEEE and the ACM. Contact him at michael.sheng@adelaide.edu.au.

Schahram Dustdar is a full professor of computer science (informatics) and he heads the Distributed Systems Group at the Vienna University of Technology. His work focuses on Internet technologies. Dustdar is a member of the Academy Europeana, an ACM Distinguished Scientist, and recipient of the IBM Faculty Award 2012. Contact him at dustdar@dsg.tuwien.ac.at; dsg.tuwien.ac.at/.

cn *Selected CS articles and columns are also available for free at http:// ComputingNow.computer.org.*