# Web of X Service Environment for Ubiquitous Computing

Zhenyu Wu, Chunhong Zhang, and Yang Ji

Mobile Life and New Media Lab,
Key Lab of Universal Wireless Communications, Ministry of Education,
Beijing University of Posts and Telecommunications,
Xitucheng Road 10, 100876 Beijing, China
{shower0512,zhangch,jiyang}@bupt.edu.cn

**Abstract.** A central concern in the area of ubiquitous computing has been the integration of digital resources including devices and heterogeneous networks with the physical world and people. However, current systems are independent to each other, which means that resources for ubiquitous computing are separated and applications could not be created on cross-platform. The openness, simplicity, flexible and standardization of Web have given insights of building up a more uniform, open and scalable service environment for ubiquitous computing. Therefore, we propose the concept of Web of X (WoX) Service Environment for ubiquitous computing in this paper. In this article, we analyze the requirements of building a service environment for ubiquitous computing, and then the conceptual architecture of WoX Service Environment and some key technical solutions are proposed as well.

**Keywords:** Ubiquitous Computing, Web, REST, API, Mashup.

## 1 Introduction

A central concern in the area of ubiquitous computing has been the integration of digital resources including devices and heterogeneous networks with the physical world and people. In particular, the "Internet of Things (IoT)" has essentially explored the development of applications built upon various networked physical objects [1]. However, current related researches mainly focus on building applications and vertical systems based on specific scenarios and industries, or considering how to leverage current broadband wired or wireless access technologies to interconnect everyday object, which means that the applications are usually built upon independent systems, resources could not be shared and efficiently utilized among different infrastructures and even the development of pervasive applications becomes tedious for common users for the reason that most of interfaces in a system are defined proprietarily and there is a lack of clear, standardized, and interoperable communication protocols that can be understood by fridge, TV set, sensors and networks, so a uniform service environment which is cross-industry, cross-network, cross-terminal and opened to users to create personalized applications is necessary.

With Web 2.0 applications the focus is on the user and user-generated content on the one hand, and on the other hand on a set of technologies (e.g., AJAX, RSS) that support the development of highly interactive interfaces that offer a rich user experience, similar to common desktop applications. As shown by the unparalleled scalability of the Internet, simple technologies (e.g. HTTP) can give birth to very efficient and flexible systems, where a large variety of hardware and software platforms coexist and interact smoothly. Open access to data through services on the Web has enabled information to be reused across independent system, therefore has lowered the access barrier that allows people to develop their own applications. Furthermore, the development of composite applications on top of the open and simple standards that made the Web so successful (REST, XML, HTTP, or Atom) to interconnect physical devices which called "Web of Things (WoT)" has now become another trend in the era of IoT and ubiquitous computing, which has given a vision for building a more uniform, open and scalable service environment for ubiquitous computing.

Based on these observations, we propose to leverage the existing and ubiquitous World Wide Web (WWW) as common ground where everyday devices and networks could interact with each other, and a user-centric Web of X (WoX) Service Environment for ubiquitous computing is also proposed in this paper. The X in WoX means heterogeneous resources surrounding persons for ubiquitous computing, such as devices, network resources, as well as current Web resources in Internet, and WoX Service Environment aims to facilitate efficient prototyping ubiquitous computing application and enable users to generate service by themselves according to their requirements and preferences by integrating the capabilities of heterogeneous network and distributed devices (mobile phone, sensors, appliances and etc.) via Web technologies.

The remainder of this paper is organized as follows: Section 2 summarizes related work. Section 3 describes the paradigms and requirements of designing a uniform service environment for ubiquitous computing. Section 4 proposes the conceptual architecture of the WoX Service Environment. Section 5 discusses the key technical issues related to constructing the service environment. Finally some challenges for future study are presented in the last section.

## 2 Related Work

A lot of work has proposed ideas to construct a cross-platform environment to eliminate the barriers between heterogeneous network infrastructures and different terminal platforms.

MUSE is proposed by Ji Yang et.al in [2] as a vision for future service and architecture for ubiquitous networking and computing. MUSE brings in the concept that enabling Always Best Experience (ABE) to users in terms of context awareness, flexibility and ubiquitous intelligence, through cooperation among various heterogeneous wire or wireless networks and devices. The essence of MUSE is to abstract the capabilities of both networks and terminal devices and specific services could seamlessly compose suitable capabilities to deliver to end users according to the requirements. Though the concept of MUSE brings in a vision of the trend of future ubiquitous computing, however, what exact technologies could be utilized in this concept is not discussed yet.

With the development of Web technology, accessing non-web-enabled to the WWW is not a new idea yet, while more and more researches have focus on aggregating heterogeneous resources into the Web and letting Web act as a ubiquitous hub to interconnect devices, network infrastructures and applications, which gives a technical option for MUSE. For example, webinos [3] project defines and delivers an Open Source Platform and software components for the Future Internet in the form of web runtime extensions, to enable Web applications and services to be used and shared consistently and securely over a broad spectrum of converged and connected devices, including not only mobile phones, but also PC, home media (TV) and in-car units, as well as to offer a common set of APIs to allow applications easy access to cross-device functionality. Nevertheless, the motivation of webinos is to enable web application to run on different kinds of devices and access the underlying resources in uniform standardized interfaces, however, the resources of devices could still not be linked, shared and reused through Web into Internet.

Thus, some researches are considering linking the Web and physical objects. For example, [4] put forward the concept of “Web of Things”: propose to reuse and adapt patterns commonly used for the Web, and introduce architecture for the Web of Things [5]. Their research is focused on the development of composite applications on top of the open and simple standards that made the Web so successful (REST, XML, HTTP, or Atom) to interconnect physical devices. They embed Web servers [6], [7], [8] on smart things and apply the REST architectural style [9], [10] to the physical world, so the resource on the smart things could be opened and shared into Internet.

Furthermore, a number of implemented systems have already shown some features that integrating diversities of resources and being opened to the public as open APIs via web standard. For example, Pachube [11] is a global centralized website which could store, share/discover real-time sensor, energy and environment data from objects, devices & buildings around the world. It defines a uniform REST web API to gather third party resources and open to the public to implement applications, and also it defines an open data exchange format protocol which is called EEML[12] to guarantee that heterogeneous datum are packaged and exchanged in a uniform pattern. SenseWeb [13] is another unified system developed by Microsoft which could collect, share, process, and query sensory data from the globally shared sensor network. It aims to provide an open architecture such that third parties can easily register sensors or repositories of sensory data to contribute as part of SenseWeb by providing Sensor Gateways that provide a uniform interface to all components above it to get all sensors connected, and other components of SenseWeb access the gateway to obtain sensor data streams, to submit data collection demands, or access sensor characteristics through a standardized web service API. The gateway implements sensor specific methods to communicate with the sensor. Meanwhile, SenseWeb also enables third parties to easily develop sensing applications that use the shared sensing resources provided by SenseWeb. These approaches are based on a centralized repository and devices need to be registered before they can publish data, thus are not sufficiently scalable and are more concerned with data storage and retrieval.

## 3 Paradigms Required and Requirements Analysis

To design a user-centric uniform service environment for ubiquitous computing, it is necessary to analyze the requirements from both the user's and service's points of view.

What users concern about is how to access to a target service with the best user experience which could meet their requirements with the dynamic variance of the context and scenario. In a further stage, users are even not only the consumer of the service, but they could also act as a creator and manager of the service, which means that users could efficiently prototype new services within their preferences in different scenarios, and conveniently deploy them with as few configurations as possible.

While from the perspective of service, with the evolution of network technologies, such as 3G/4G and the increase of diversities of terminal platforms, all services are required to be delivered via a uniform platform which need not care about the details of heterogeneous terminals and networks and services could acquire the terminal capabilities and network capabilities through standardized open interfaces provided by the platform, so services provider or end users can develop and deploy more high quality services more flexibly, and service could be customized more dynamically.

Therefore, to feed the requirements above, several paradigms are required:

**Paradigm 1:** (the most important) the user-centric service environment for ubiquitous computing should be based on Web. The development of Web has shown the success of it, and previous researches have proven that Web is simple, open, standardized and flexible. Moreover, the interaction style of Web is accepted by more and more common users, and the development of web application becomes more and more convenient for developers than other desktop and mobile platforms, which means users could rapidly prototype personalized applications and build up a healthy ecosystem. So Web should be a first option to integrate heterogeneous platforms and different current technologies, and all the terminal, network and other computing resources should be abstracted as web resources.

**Paradigm 2:** User could share, manage and even deploy resources/services in the service environment. Web 2.0 and social network service has reveals the power of the involvement of end users into the sharing and innovating activities. More high-quality data and fascinating application appears. In pervasive computing, the more context data about specific user could share with others and the fewer configurations for users, the more customized applications could be created and the more possible the user's requirements could be met.

**Paradigm 3:** The architecture should be as flexible and scalable as possible. It is not only serving local area or specific scenario but also should support wide-area services and scenarios.

**Paradigm 4:** Security and privacy must be guaranteed. In ubiquitous computing, there must some personal and private information, so to prevent this information being stolen and opened to public must be guaranteed.

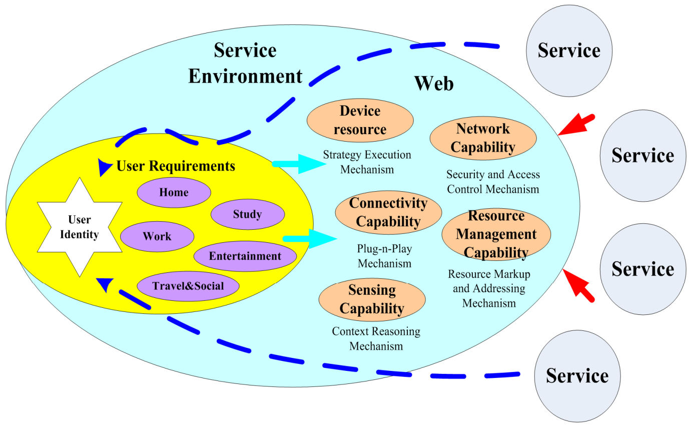

**Fig. 1.** Requirements analysis of user-centric uniform service environment for ubiquitous computing from both user's and service's points of view

According to the design paradigms, several specific technical requirements should also be met:

**Req 1:** Follow RESTful-stlye architecture design principle. The essence of REST is to focus on creating loosely coupled services on the Web so that they can be easily reused [14]. REST is actually core to the Web and uses URIs for encapsulating and identifying services on the Web. Resources should be available through a uniform interface with well-defined interaction semantics, as is *Hypertext Transfer Protocol (HTTP)*. For data exchange and resource description, On the Web, media type support in HTTP and the *Hypertext Markup Language (HTML)* allow peers to cooperate without individual agreements. For machine-oriented services, media types such as the *Extensible Markup* Language *(XML)* and *JavaScript Object Notation (JSON)* have gained widespread support across services and client platforms. Moreover, the interaction should be kept stateless which requires requests from clients to be self-contained, in the sense that all information to serve the request must be part of the request. HTTP implements this constraint because it has no concept beyond the request/response interaction pattern; there is no concept of HTTP sessions or transactions.

**Req 2:** Device and networks should access to Web. To make sure that the service environment is based on the Web and the capabilities provided by the service environment could be invoked in Web protocols, the devices and networks which are not support Web must be extended to support accessing to Web.

**Req 3:** The resource of every object could be identified, named and addressed in a standard way. All the capabilities of objects such as terminal and network components are abstracted as web resources, so the service environment should name all the resources with unique URI and provide a mechanism to address them.

**Req 4:** There should be a uniform resource or service model. A model/framework to describe resources and their relationships between them is necessary.

**Req 5:** Support Plug-and-Play (PnP). To ease the cost of device deployment for end user, the service environment should provide some mechanism for automatic resource discovery and configuration on devices with as fewer interferences as possible for common users.

**Req 6:** Support resource access control, authorization and authentication. To guarantee secure access to the private or authorized resources provided by terminals and networks, the service environment should provide some authorization and authentication mechanisms to make sure that every resource is in a secure environment.

**Req 7:** Enable context-aware. To provide more intelligent and personalized service, service environment should provide some context-aware computing capabilities based on semantic and ontology, such as context modeling, context storage, lookup and reasoning.

**Req 8:** Provide a uniform user interface and engine for service mashup. To facilitate the quick prototype of customized application for non-tech users, service environment should provide an intuitive interface for user to create their own composite service in a drag-and-pull way, and a mashup engine is also necessary to support service orchestration based on the Web Service capabilities provided by terminals and networks.

## 4 Web of X Service Environment Architecture

In this section, we have proposed a conceptual architecture for Web of X Service Environment for ubiquitous computing according to the requirements analyzed in the previous section.

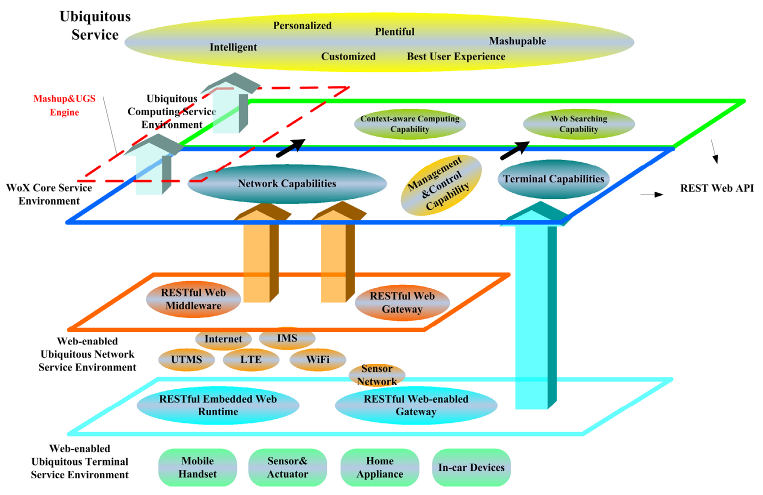

**Fig. 2.** Web of X Service Environment Architecture

The **WoX Service Environment** consists of four main components as Figure 2 shows: web-enabled Ubiquitous Terminal Service Environment (wUTSE), web-enabled Ubiquitous Network Service Environment (wUNSE), Web of X Core Service Environment (WoXSE) and Ubiquitous Computing Service Environment (UCSE).

**wUTSE** is a service environment surrounding terminals with web access which may include web-enabled devices (mobile handsets) with an embedded web runtime and non-IP devices (sensors, home appliance or in-car devices) brokered by web-enabled gateways. It may be offered by single terminal equipment or a set of networking distributed components.

**wUNSE** is a service environment based on heterogeneous networks which cover from wired network to wireless network, from IP network to non-IP network, from access network to core network, from telecom domain to Internet domain, such as Internet, IMS, UMTS, LTE, WiFi and even sensor network. wUNSE bridges the components of networks into the WWW by web middleware or web proxy, so the capabilities of different networks could be exposed as Web Service in uniform Web standard.

The wUTSE and wUNSE shield the details of internal architecture of terminals and networks, and the connectivity between them. Moreover, the wUTSE and wUNSE also both offer services the information about terminal and network capabilities in a REST way, which means that the capabilities of terminals and network will uniformly be abstracted as Web resources with unique identities (URI) and could be accessed via HTTP in a stateless client-server way with uniform method (GET, PUT, POST and DELETE).

**WoCSE** is a service environment which acts as a control components for all the exposed REST web service resources. The capabilities of networks and terminals are all opened as a REST web API and abstracted as web resources, so WoCSE should be responsible for managing these web resources, such as web service resource registration, update and discovery within global area, and data storage and management, as well as scheduling of service and authorization/authentication. The same as wUTSE and wUNSE, the capabilities of WoCSE are opened a REST Web APIs and could be invoked by applications as well.

**UCSE** is a service environment for context-aware computing and Web searching to support intelligent and personalized services. Resources exposed from wUTSE and wUNSE are all service context information surrounding end users, so UCSE is responsible for context data modeling, storage and reasoning by invoking web services provided by wUTSE and wUNSE to retrieve raw date of context information from devices and networks. There may exist diversities of context-aware computing engines and semantic web searching engines which are built upon cloud computing infrastructures according to different scenarios. Also these computing capabilities are opened as REST Web APIs and could be invoked by services.

Furthermore, as a user centric service environment, WoX Service Environment also provides some kind of User Generated Service (UGS)/Mashup engine for both tech-savvy users and non-tech users to create composite services by their own need and preference based on the Web API provide by the devices and networks. The UGS/Mashup engine could provide an intuitive interface for user to drag-and-pull components to generate target services at front end and process dynamic service orchestration at back end, so that users only need only concern about the logic of the service.

## 5 Key Technical Issues Discussion

For WoX Service Environment, there remain some key technical issues that should be discussed, such as how to access resources on non-web devices into Web and how to discovery resources based on WoX Service Environment, and some solutions should be proposed as a reference to implement and deploy the WoX Service Environment.

### 5.1 Resource Access

To build up a Web-oriented service environment, it is necessary that the non-web resources, especially the resources on the non-web devices, such as mobile phones, sensors, actuators, and home appliance could be accessed via Web interface. In Fig 3, two ways to get non-web devices' resources accessed via RESTful Web interface are proposed as follows:

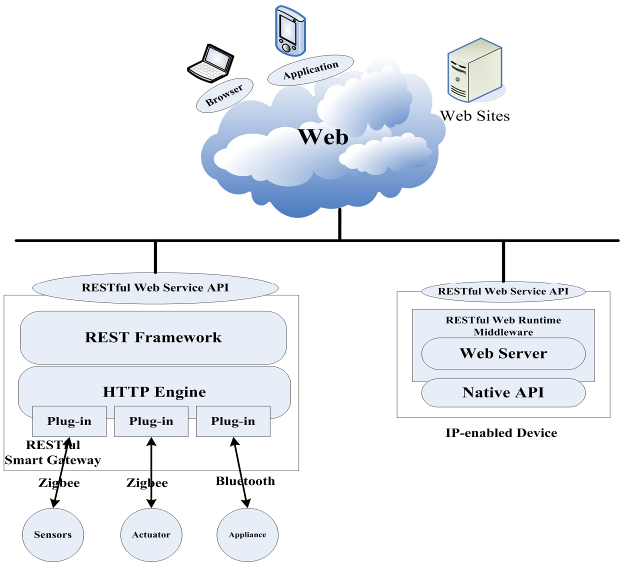

**Fig. 3.** Two ways to get heterogeneous resource accessed via Web

Most embedded devices do not always have an IP (Internet Protocol) address and are thus not directly addressable on the Internet. However, it is very likely that more and more real-world devices will become IP-enabled and have embedded HTTP servers (in particular with 6LowPAN), making them able to understand the Web languages and protocols [15] [16]. Such Web-enabled devices can be directly integrated and make their RESTful APIs directly accessible on the Web. This integration process is shown on Fig 3: each device has an IP address and runs a Web Server on top of which it offers a RESTful API to the mashup developer, and we call it **RESTful Web Runtime Middleware**.

While such Web-enabled devices are likely to be widely spread in the near future, direct integration of real-world devices into the Web is still a rather cumbersome task. In particular, when devices do not support IP or HTTP as is usually the case with wireless sensor networks (WSN), a different integration pattern is needed. So we propose to use the concept of **RESTful Smart Gateways** as intermediate element that bridges the Web with devices that do not talk IP. The RESTful Smart Gateway is composed of three main components. The Plug-ins acts as drivers to discover the devices on a regular basis by scanning the environment in other communication protocols, and for each kind of communication there is a plug-in, such as plug-in for Bluetooth API and plug-in for Zigbee APIs, so the Smart Gateway could directly access data via native requests of different proprietary communication protocols. The HTTP engine acts as an embedded web server which handles HTTP requests and responses. The RESTful Environment acts as a REST framework to support REST operations, and it could map the URLs to the resources the devices possess and provide uniform interfaces (HTTP Method: GET, PUT, DELETE, POST) to interact with the resources.

## 5.2 Resource Discovery

To lower the cost of deployment and management for end user, the WoX Service Environment should be as free as possible from explicit human administration. This is especially challenging in ubiquitous computing, where devices can dynamically enter and leave the network. If all the devices require explicit configuration to work together, the burden of administration quickly overwhelms any potential benefit, so some kind of PnP mechanism is required to eliminate the burden of administration.

In the vision of WoX Service Environment, the capabilities of global networks and devices are abstracted as RESTful web resources which are identified by HTTP URI, so a large-scale, multi-domain service infrastructure requires a resource discovery system that is open, scalable, robust and efficient to a greater extent than a single-domain system. For existing discovery system, two common models are usually adopted. One is non-directory based way; the other is directory-based way. For globally resource discovery, the directory-based way is the most suitable model [17].

Therefore, in the WoX Service Environment, a directory-based resource discovery model is proposed. It is assumed that the resources are distributed into different domains according to specific features, such as location, type of device, etc. and there is unique directory in each domain which is responsible for registration, update and discovery of resource in the domain. Furthermore, every identified resource should be described in a uniform *descriptive record* to describe its address (URL) and its attributes, such as its location, type, owner and the method to invoke it. In the proposed model, the whole framework is divided into three logic entities:

**Resource Agent (RA)** is a logical entity that possesses web resources on smart objects, such as web-enabled devices and smart gateway. In each RA, its resources will be identified with URLs as a RESTful web service interface.

**WoX Proxy/Directory Agent (WP/DA)** is a logical entity that maintains resource descriptive information and processes queries and announcements from RAs and clients, and each WP/DA will be responsible for managing all the RAs in its local domain. The topology of the WP/DA could be hierarchical or Peer-to-Peer structure, and the descriptive information of registered resources will be maintained and synchronized among these WP/DAs in certain mechanism.

**Bootstrap Agent (BA)** is a logical entity where clients and services attempt to initiate the discovery process via establishing the first point of contact within the system. With initial configuration, the BA will manage lists of all WP/DAs' information including the address and the domain information in its charge.

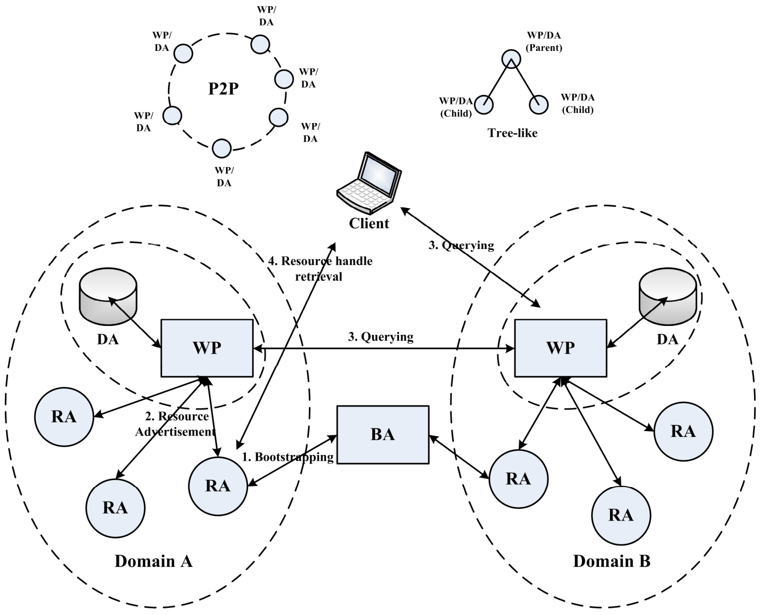

**Fig. 4.** Directory-based resource discovery model in WoX Service Environment

Specifically, the process of the resource discovery could be viewed in four steps as Figure 4 shows:

•**Bootstrapping:** the RA will query BA to find the local or the nearest WP/DA and BA will return the address of target WP/DA that is responsible for its registration.

•**Resource Advertisement:** the RA advertises itself by registering with the target WP/DA a *description file*. This record follows the XML schema and represents a description of the attributes of the resource and contains series of operations to that resources as well as the URL of the resource.

•**Querying:** the client will look for a desired resources. The querying process is handled by the WP/DA with which a RA is associated; the client may first ask any

WP/DA to find out which WP/DAs provide a desired resource, i.e. the resource that matches the *description file* provided by the client, and finally select one and ask its related WP/DA to retrieve the *description file* from it. The lookup process among WP/DAs could be based on a tree-like hierarchical structure or P2P structure [18]. If the attributes of the resource is refreshed, the WP/DA can periodically check the availability of the resource and update its *description file*.

•**Resource Handle Retrieval:** the client receives the *description record* of desired resource and access to the target URL via RESTful web service API provided by the RA.

## 6 Challenges for Future Study

As a design for future service and architecture, there are still lots of open issues in the WoX Service Environment design for further investigations.

In WoX Service Environment, the capabilities of devices and networks are abstracted as Web resources in REST way; however, a uniform resource representation framework is still needed to describe heterogeneous resources. A semantic-based or ontology-based method might be adopted that not only human could understand the language but also could machine understand to support automatic and intelligent resource discovery and configuration.

Security is always a critical factor in system design. Without the guarantee by an all-around and powerful security mechanism, any system will become far from practical services and applications. Since existing security mechanisms are designed for certain kinds of applications, networks and terminals, although fulfilling the tasks well in their fields, they are definitely insufficient when dealing with the complicated heterogeneous network environments, varied services and terminal environment in the future, where the access control of resources on different devices and networks, delegated authentication mechanism for third party mashup should be considered much. Therefore, a customized, complete and effective security architecture is very important and absolutely necessary for this open, cross-platform, heterogeneous and multi-serviced environment of WoX.

Moreover, since WoX Service Environment introduces more decentralized and self-organized paradigms in resource discovery, allocation and management, the peer-to-peer mode service provisioning should be supported in the whole architecture, not only in connectivity aspects, but also in operational and business aspects. There is some peer-to-peer solution available already. Therefore, how to utilize these peer-to-peer solutions into the WoX Service Environment needs to be well studied in the future.

## 7 Conclusion

In this paper, we propose that Web technologies are — contrary to popular belief — a suitable protocol for constructing a uniform service environment for ubiquitous computing and building applications on top of services offered by different devices and heterogeneous networks. After summarizing the core design paradigms and requirements of the future ubiquitous computing service environment, we have proposed a conceptual architecture for the WoX Service Environment based on RESTful architectural principles. Then several key technical issues in constructing the WoX Service Environment, such as resource access framework and resource discovery mechanism, are discussed and relative solutions are also proposed. Finally, some challenges, such as the resource representation model, security mechanism, as well as peer-to-peer solution are also discussed for future study.

**Acknowledgments.** The work is supported by the International Science and Technology Cooperation Plan Project (No. 2010GR0002) and National Science and Technology Major Project (No. 2011ZX03005-004-04) and Fundamental Research Funds for the Central Universities (No. 2010PT17).

## References

1. Fleisch, E., Mattern, F.: Das Internet der Dinge, 1st edn. Springer (July 2005)
2. Yang, J.I., Ping, Z., Zheng, H., Xu, W., Yinong, L., Xiaosheng, T.: Towards mobile ubiquitous service environment. Wireless Personal Communications 38(1), 67–78 (2006)
3. webinos, `http://webinos.org`
4. Guinard, D., Trifa, V.: Towards the Web of Things: Web Mashups for Embedded Devices. In: 2nd Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009), Madrid, Spain (April 2009)
5. Guinard, D., Trifa, V., Wilde, E.: A resource oriented architecture for the web of things. In: Proc. of IoT 2010 (IEEE International Conference on the Internet of Things) (November 2010)
6. Hui, J.W., Culler, D.E.: IP is dead, long live IP for wireless sensor networks. In: SenSys 2008: Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems, pp. 15–28. ACM, New York (2008)
7. Duquennoy, S., Grimaud, G., Vandewalle, J.-J.: The Web of Things: interconnecting devices with high usability and performance. In: 6th International Conference on Embedded Software and Systems (ICESS 2009), Hangzhou, Zhejiang, China (May 2009)
8. Guinard, D., Trifa, V., Pham, T., Liechti, O.: Towards Physical Mashups in the Web of Things. In: Proceedings of the 6th International Conference on Networked Sensing Systems, INSS 2009 (2009)
9. Fielding, R.T., Taylor, R.N.: Principled design of the modern web architecture. ACM Transactions on Internet Technology 2(2), 115–150 (2002)
10. Richardson, L., Ruby, S.: RESTful Web Services. O'Reilly (2007)
11. Pachube, `http://www.pachube.com`
12. Extended Environments Markup Language (EEML), `http://www.eeml.org`
13. Kansal, A., Nath, S., Liu, J., Zhao, F.: SenseWeb: an infrastructure for shared sensing. IEEE Multimedia 14(4), 8–13 (2007)
14. Pautasso, C., Wilde, E.: Why is the Web Loosely Coupled? A Multi-Faceted Metric for Service Design. In: Proc. of the 18th International World Wide Web Conference (WWW 2009), Madrid, Spain, pp. 911–920 (2009)
15. Dunkels, A., Vasseur, J.: Ip for smart objects alliance. Internet Protocol for Smart Objects (IPSO) Alliance White paper No.2 (September 2008)
16. Hui, J., Culler, D.: Extending IP to Low-Power, wireless personal area networks. IEEE Internet Computing 12(4), 37–45 (2008)
17. Zhu, F., Mutka, M., Ni, L.: Service Discovery in Pervasive Computing Environments. IEEE Pervasive Computing 4, 81–90 (2005)
18. Stoica, I., et al.: Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications. IEEE/ACM Trans. Net. 11(1), 17–32 (2003)