A Middleware-Based Network Architecture for the Web of Things

Jing Sha, Yang Ji
Beijing University of Posts and Telecommunications
Beijing, China
Email: wondersha2008@gmail.com

***Abstract*—Web of Things has been proposed to facilitate the integration and composition of everyday device like sensors and actuators with existing standards and blueprints. However, the existing researches do not address all the problems such as integrating legacy devices with constrained capabilities, managing the resources with dynamic attributes. In addition, considering the large-scale deployment of devices in WoT, the deployment expense is supposed to be controlled. In this paper, we propose a middleware-based network architecture for the web of things. Within the architecture, we deal with the practical challenges in deployment. A SMART HOME system is developed as the implementation to verify and discuss the proposed architecture.**

Keywords-Web of Things; resource integration; resource life cycle.

I. INTRODUCTION

The embedded technology and wireless network has developed so much that computing has become more ubiquitous. Many applications have been developed for people to requiring data from devices like sensors and cameras. This trend has driven the development of Internet of Things (IoT) which makes things accessible through internet. In the IoT, different manufactures and operators develop their own system, which makes the data sharing and collaboration difficult. As a result the concept of Web of Things [[1], [2]] has been proposed to abstract the things into web resources.

However, current WoT solutions focus more on the up layer research by adapting existing web technology to the WoT scenarios. In fact, in the SmartBUPT [3] system we implement earlier, we find that there are still some practical issues when implementing the concept. For example, large amount of things are legacy devices, which possess constrained capabilities compared with the traditional web servers. In addition, it is necessary to manage the status of running resource after the resource discovery process.

In order to tackle these problems, we propose a middleware-based network architecture for the web of things which can be applied to deploy devices in large scale. In addition, we have realized the practical problems and consider them in the designation.

The rest of the paper is organized as follows: Chapter II presents the challenges when implementing the WoT. Chapter III introduces the architecture and solutions to practical problems. In chapter IV, we introduce the implementation of our architecture in a SMART HOME. Then the concept and architecture are analyzed and discussed in section V. Finally, conclusion and future work are given in the last section VI.

II. CHALLENGES

In order to facilitate the deployment and integration of large amount of things, standard solutions like WSDL [11] and WADL [12] have been proposed to reduce the cost of connecting and interoperating. However, such standard protocols are usually applied to the traditional web resources that are hosted on servers with strong capabilities. In contrast, resources are hosted on constrained devices like sensors and actuators in the web of things. It would be difficult to implement complex protocol directly on them. Meanwhile, things in the WoT are highly dynamic and unstable due to their physical attributes. Consequently, there are several practical problems when integrating and managing WoT resources.

A. Device accessibility

In the process to integrate things in the WoT, the primary task is to make various devices accessible. Many researches in the Internet of Things ignore a practical problem that many legacy devices do not process the basic communication and computing capabilities. It would be necessary to extend these capabilities for the legacy devices. In addition, it is also essential to reduce the use expense with a uniform access interface. This would require the translation from specified data format to a uniform one. It is fortunate that solutions like EEML [13] and SensorML [14] have provided us reference. However, it is still necessary to make some adaption to meet the WoT requirements.

Besides, the accessible device also have to provide stable data sources. Influenced by the physical attributes, resources in the WoT cannot be as stable as traditional web resource. Such instability include Mobility: the access address of resources may change when hosting devices move about; Reachability: the hosting device may be located in a local network and cannot be reached directly from public network; Resource constraint: the performance of resources can be influenced by limited power or bad network condition. In fact, many solutions like Reverse Http [8], Mobile IP [9] and dynamic DNS [10] have been proposed to tackle such problems.

B. Resource exposition

Compared with Internet of Things, the WoT concentrate more on the explosion of resource. There are several web protocols such as SOAP and Restful [4] Web Service that can be used for exposition, mashup and integration of things. To support such protocols, the things have to process the capabilities to deploy embed server on themselves. It is rather difficult for many legacy devices. Some systems like cosm [5] propose to collect the data to a central repository and then expose them. However, such method will increase the delay of acquiring real-time data. In addition, storing the data in remote location also results security problems.

Another challenge that the WoT has to deal with is the resource search. On one hand, a search service should be provided to the users to filter the resources matching requirements. On the other hand, the access address of resources should also be provided. Considering the dynamic character of WoT resources, it is essential to maintain the availability of resource.

C. Life cycle management of resource

In the WoT, maintaining the status of resource is quiet important. The status can be changed as hosting devices join, move and leave. Firstly, the process of device joining is supposed to include device discovery and registration. Some existing research like UPnP [6], JiNi [6] have focused on this process. Other propositions like HTTP compression [7] also provide reference in the WoT context. However, in some application scenarios, we find that such protocols are too complex for implementation and a lighter protocol is more suitable. Secondly, the running status of resource is supposed to be monitored after devices have joined. Status like normal, suspended and left should be maintained according to different situations.

In order to deal with such problems, we propose a middleware-based network architecture for the web of things. We also implement the architecture in a SMART HOME system for demonstration. It is detailed in the following sections.

III. Middleware-Based Network Architecture

The Middleware-Based Service Discovery Architecture for the Web of Things is shown in Figure 1. There are two key components which are presented as gateway middleware and Resource Management Center (RMC). The gateway middleware is the software that can be deployed on any computer with qualified operating system. It is used in a local network to manage all the connecting devices like sensors or actuators and make them accessible through web service. RMC manages resources in global context by synchronizing with gateway middleware. Consequently, based on the service provided by the RMC gateway middleware, the WoT applications have access to the resource matching requirements. In the following subsections, each key component of the architecture will be detailed.

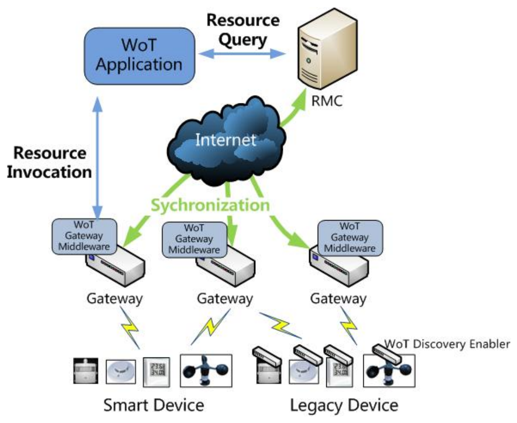

Figure 1. Architecture Overview

A. WoT Discovery Enabler

In our implementation scenario, many devices like sensors and actuators only have their dedicated function like monitoring the temperature. In order to connect such legacy devices to the network, as shown in Figure 2, we design the WoT Discovery Enabler (WDE) as the component which provides the devices with necessary communication and storage capabilities. The component communicates with devices through serial ports, which is commonly supported by legacy devices according to our research. When it connects with devices, the WDE can provide following capabilities:

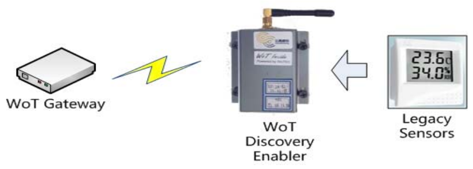

Figure 2. WoT Discovery Enabler

IP-based communication: the component implements IP communication protocol and can also support WiFi access. **Storage**: the component is designed to possess storage space, which can be used to store device profile and device registration information. The content stored can be rewrite or delete according to requirement. **Discovery protocol**: the component can be developed to support light discovery protocol. For demonstration, we implement a simple discovery protocol over it.

B. WoT Gateway Middleware

Integrating things on large scale is one of the most important requirements in the Web of thing. On one hand, many legacy devices do not possess the capability to support an embedded server to expose itself to the web. On the other hand, deploying with dedicated gateway would be of high cost. To tackle this problem, we propose the WoT gateway middleware which can be deployed on any computers with qualified operating system. A typical application scenario is that the middleware can be deployed on a PC and then the PC can be considered as a gateway. User can have access to the devices (e.g. sensors and lights) connecting to the PC through web service. We have applied this middleware in our SMART HOME system. Figure 3 shows the software building blocks of our gateway middleware.

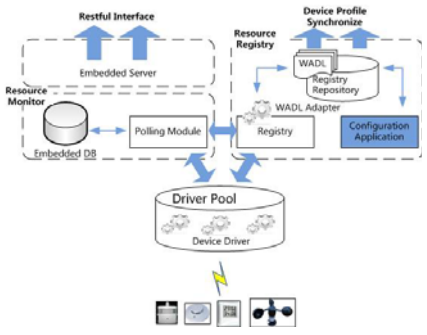

Figure 3. WoT gateway middleware framwork

Driver Pool provides various device drivers when the gateway middleware communicate with the connecting devices. When certain type of device connects, the gateway middleware can dynamically load the driver if it is available. Otherwise, the middleware can download new drivers from the RMC. Thus the devices can be plug in and out in running time.

Resource Registry is responsible for the management of resources life cycle. The stages of resource life cycle include Connected, Online, Suspended and Left. When a new device joins the local network with a gateway middleware, the first step to connect to gateway is registration. This process is shown in Figure 4. The gateway middleware broadcast the "detect" massage in the local network and the incoming device responds with the registration information stored in the WDE. After verifying the information, the gateway middleware responds with a resource identifier to the WDE. The WDE can store the identifier and use it every time it communicates with gateway middleware. Upon the registration process is completed, the life status of the newly joining resource is set to Connected.

In the Resource Registry, the profile of resource is maintained in the Registry Repository. The registration profile sent by the WDE include resource type, geographic location, resource description as well as operation list which clarifies the operation provided by the devices. In order to facilitate the integration of such resources, we translate the profile into standard format before storing them. For example, the operation list is translated into WADL, which can be compatible in the discovery of restful web service. All the information in the repository is synchronized with the RMC periodically.

The Resource Monitor is responsible for both the data caching and resource status updating. Upon the resource has registered to the gateway, the data polling component begins to request the resource for real-time data periodically. The data is then stored in the embedded DB as cache. Considering the communication with device may results in considerable delay due to bad network condition, the cache is necessary especially in time-sensitive scenarios.

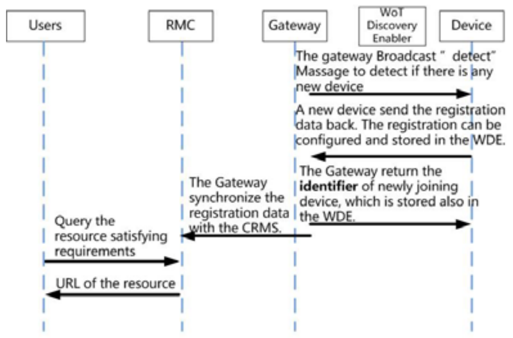
Users
RMC
Gateway
WoT Discovery Enabler
Device
The gateway Broadcast " detect" Massage to detect if there is any new device
A new device send the registration data back. The registration can be configured and stored in the WDE.
The Gateway return the identifier of newly joining device, which is stored also in the WDE.
The Gateway synchronize the registration data with the CRMS.
Query the resource satisfying requirements
URL of the resource

Figure 4. New Reource Joining Process

Meanwhile, we acknowledge that the working status of service provided by the device can be updated depending on whether there is response to the polling request. If resources can response to the polling normally, the status of resources are set to Online. If there is no response from certain resource, the data polling component notifies the Registry to update working status of the device. As shown in Figure 5, the gateway middleware broadcast the "check" massage and records the resource that does not answer to the "check" massage. The resource status will be set to Suspended. After a fixed interval, a second "check" massage is broadcast; the status of a Suspended resource will be set to Left if there is still no response. Any normal response before the second "check" massage will reset the status to Online. Also the updating resource status is synchronized with the RMC.

Over all the components inside the gateway middleware, an embedded server is used for resource explosion. We use restful web service to expose the resources to the web. User can query the RMC for the URL of each resource and invoke the resource through URL. The uniform access interface facilitates the mashup and integration of resources.

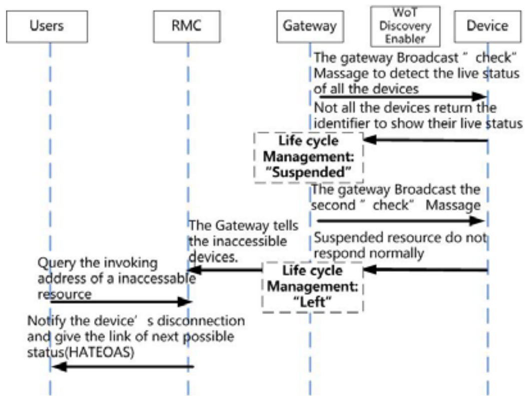
Users
RMC
Gateway
WoT Discovery Enabler
Device
The gateway Broadcast " check" Massage to detect the live status of all the devices
Not all the devices return the identifier to show their live status
Life cycle Management: "Suspended"
The gateway Broadcast the second " check" Massage
The Gateway tells the inaccessible devices.
Suspended resource do not respond normally
Query the invoking address of a inaccessable resource
Life cycle Management: "Left"
Notify the device' s disconnection and give the link of next possible status(HATEOAS)

Figure 5. Resource Leaving Process

C. *Resource Management Center*

The RMC is implemented as a server that maintains the profile of resources. When it communicates with gateway middleware, RMC acts as a central storing server that stores the synchronized profile sent by each gateway periodically. The profile includes registration information, addressing information and resource status.

Besides, the RMC provides the discovery service to the WoT application. In the query process, the RMC works as the query and addressing server. When receiving a query request, the RMC will filter the resource profile by the request and then response with the result list. The result includes service description and service working status. The service description is described with WADL and the address of service is represented as an URL which can be directly invoked using the HTTP request. As shown in Figure 6, the RMC consists of several parts:

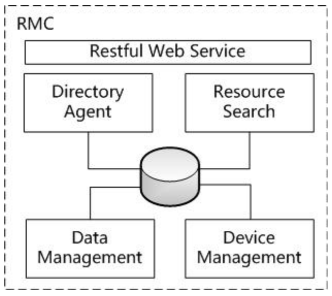

Figure 6. Resource Management Center Framework

1) Directory Agent

The Directory Agent maintains the access address of each resource. In our demonstration system, each resource is exposed through restful web service and thus identified as an URL which directs to the address of the a gateway. However, as the devices may move about, the URL may change when devices connect to different gateways and the invocation through the old URL may fail. So the Directory Agent is designed to map the URL to each resource dynamically and response it to the user to ensure the availability of each URL.

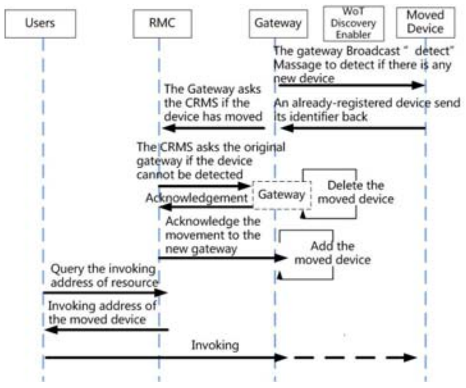
Users
RMC
Gateway
WoT Discovery Enabler
Moved Device
The gateway Broadcast " detect" Massage to detect if there is any new device
The Gateway asks the CRMS if the device has moved
An already-registered device send its identifier back
The CRMS asks the original gateway if the device cannot be detected
Acknowledgement
Gateway
Delete the moved device
Acknowledge the movement to the new gateway
Add the moved device
Query the invoking address of resource
Invoking address of the moved device
Invoking

Figure 7. Reource Moving Process

Figure 7 shows the dynamic mapping process. When a moved device connect to a new gateway, the gateway firstly queries the RMC that whether the new device has moved. Then RMC asks the original gateway for acknowledgement. If the original gateway has detected that the device has left its local network, it responds to the RMC with acknowledgement. At last, the RMC map the new URL to the resource.

2) Resource Search

It is expected that there would be massive amount of resources in the WoT. So it is necessary to provide the WoT users with the resource search service. In our demonstration system, users can search the resource by geographic location, resource type (e.g. temperature) and access right (e.g. public).

3) Device Management

The information maintained by the device management component includes **Device Topology**: it describes the devices managed by each gateway and is updated as the devices move. **Device status**: it describes the running status of devices and is updated when devices are online or offline. **Device Profile**: it describes the attributes like manufacture information. **Heartbeat**: it is used to monitor the status of each gateway. Meanwhile, in case that the gateway is located in the network that distributes short-term IP address periodically, the heartbeat information can be used to acquire the changing IP address.

4) Data Management

There is an option for the gateway middleware to store history data in the local system. Considering that the storage space may be limited, the gateway middleware can upload history data to RMC if necessary. The data management component can store the data and expose them through web service.

IV. IMPLEMENTATION

Based on an aging care cooperation project between China and Finland, we have proved our architecture in a SMART HOME system.

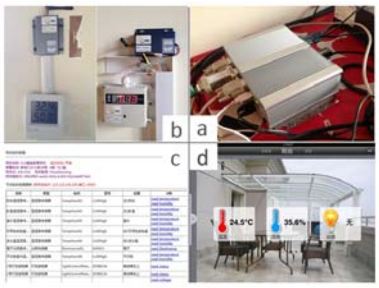

Figure 8. Overview of our demonstrator

The gateway middleware is deployed on a industrial personal computer (Figure 8a). It has a Windows operation system which is common in most home PCs. By deploying several communication modules on it, the gateway can connects home devices like sensors and actuators via WiFi, Zigbee and serial ports. Figure 8b shows the WoT Discovery Enabler deployed in our system and with them any devices can connect to the gateway in running time. Besides, we use the IIS server of the windows as our embedded server to expose our resource. The restful web service is developed based on the WCF of .net Framework 4.0. We have investigated other optional restful frameworks, such as

RESTlet framework [15] in JAVA, which can be deployed over JVM either in a Linux or Windows environment.

As an example of implementation of RMC, we use cloud service to deploy our framework, it opens service such as query, discovery and data synchronization.

As proof of concept, we develop aging care application based on our architecture. The applications include both web application (Figure 8c) and smart terminal (e.g. iPad, android devices) application (Figure 8d). All of them use the restful API from the home gateway to invoke resources.

V. ANALYSIS

Based on the SMART HOME system, we validate the technical feasibility and responsibility of the middleware-based architecture for the web of things. It also tackles the problems we proposed earlier when implement the concept of web of things.

Integration of heterogeneous devices

In the SMART HOME system, we use a middleware, which can turn the hosting devices into a gateway, to manage the heterogeneous devices inside the local network. Using the WoT Discovery Enabler, the legacy devices can plug in and out in running time. The deployment of devices can be on large scale with extension of middleware. Besides, the resources are exposed by restful web service that could be invoked by users. Such light and uniform interface reduces the use expense of resources.

Adaption to the mobility of resource

Based on the implementation of discovery protocol in the system, we have tested the process when a device switches to another access gateway. According to the log information, the switch process includes synchronization among the related gateways and RMC. After this process, the moved resource can be invoked through a new access address.

Management of resource life cycle

In our system, the life cycle is divided into 4 stages indicating different running status of resource. To some extent this designation has deal with all the situations in our application scenarios. The designation can be extended to more complex situation if necessary. With management of resource life cycle, the status of resources is timely updated and users can know about it. Such information makes it possible that user or application can make their own adaption dynamically according to the status of resources used.

VI. CONCLUSION AND FUTURE WORK

In this paper, we have introduced a middleware-based network architecture for the web of things. Compared with existing research in the web of things, the main purposes of the proposed architecture include integration the heterogeneous devices with constrained capability and dealing with the problem brought by the dynamic attributes of resources. With the cooperation between of the gateway middleware and resource management center, the WoT can be extended on large scale.

In future work, we will concentrate more on the extension of resource management considering a more complex application scenario. In addition, along with the deployment of devices in the WoT, more distributed technology will be surveyed and applied to manage massive amount resource in the future.

ACKNOWLEDGMENTS

This work was supported by the Wireless Ubiquitous Business Environment Architecture, Key Technology Research and Demonstration based on Web (No. 2012ZX03005008)，China-Finland Cooperation Project on the Development and Demonstration of Intelligent Design Platform Driven by Living Lab Methodology (No. 2010DFA12780) and project on platform, Key technology research of multi-terminal cooperative control network in wireless ubiquitous environment (No. 2011ZX03005-004-04).

REFERENCES

[1] D. Guinard, V. Trifa, and E. Wilde: A resource oriented architecture for the web of things. *In Proc. of IEEE International Conference on the Internet of Things*, Nov. 2010

[2] D. Guinard and V. Trifa: 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.

[3] Xuang Jiang, Chunhong Zhang, "A Web-based IT Framework for Campus Innovation," etcs, vol. 2, pp.90-93, 2011 Third International Workshop on Education Technology and Computer Science, 2011

[4] R. T. Fielding, “Architectural styles and the design of network-based software architectures,” doctoral dissertation, Dept. Information and Computer Science, Univ. California, 2000.

[5] Cosm, “Internet of Things Platform Connecting Devices and Apps for Real-Time Control and Data Storage,” 2012; cosm.com

[6] R. Ahmed, N. Limam, J. Xiao, Y. Iraqi, and R. Boutaba. Resource and service discovery in large-scale multi-domain networks.IEEE Communications Surveys&Tutorials, 9(4), 2007.

[7] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach and T. Berners-Lee, Hypertext Transfer Protocol -- HTTP/1.1, IETF RFC 2616, June 1999; www.ietf.org/rfc/rfc2616.txt.

[8] M. Lentczner and D. Preston. Reverse HTTP. “http://tools.ietf.org/html/draft-lentczner-rhttp-00”.

[9] C. Perkins. IP mobility support for IPv4. “http://tools.ietf.org/html/rfc59443”.

[10] P. Vixie, S. Thomson, J. Bound, and Y. Rekhter. Dynamic updates in the Domain Name System. “http://tools.ietf.org/html/rfc2136”.

[11] WSDL. “http://www.w3.org/TR/wsdl”.

[12] WADL. “http://www.w3.org/Submission/wadl/”.

[13] EEML. “http://www.eeml.org/”.

[14] SensorML. “http://www.opengeospatial.org/standards/sensorml”.

[15] RESTlet. “http://www.restlet.org”. Accessed 20 Nov 2011.