RESTful Web of Things API in Sharing Sensor Data

Lei Gao, Chunhong Zhang, Li Sun
Mobile Life and New Media Laboratory
Beijing University of Posts and Telecommunications
Beijing, China
{gaolei, zhangch, sunli} @bupt.edu.cn

***Abstract*—Recent development in the Web of Things (WoT) domain makes it possible to access sensor data via the web. While due to different structures and interfaces of the sensor networks, it's not that easy to fully integrating their data to the web then share and reuse it later. This paper introduces a platform using RESTful Web of Things API to help share sensor data. First explains the significance and advantage of adopting RESTful style API, then points out the architecture of this platform. Later proposes the RESTful API and protocols of this platform. Finally a typical application scene of this platform is described and analyzed.**

Keywords-WoT; REST; API; Sensor

I. INTRODUCTION

With the technical progress, performance of embedded devices has been raised a lot, especially that an increasing number of sensors will be supporting the IP protocol [1], so that these sensors will possess direct connectivity to the Internet and World Wide Web [2]. In addition, with the rise and development of 3G / Wi-Fi wireless access, the Internet bandwidth is increasing while the cost is dropping continuously, which makes it conditional to deploy numbers of wireless sensors to a large scale.

However, the data interfaces vary a lot between different sensors. This paper tries to describe a platform using RESTful [3] WoT API to share and reuse the data of sensors.

The proposal of Web of Things (WoT) devotes to connect every thing to the web. And web service makes it possible to open access to data. Therefore, information could be reused across independent system. This has lowered the access barrier that allows people to develop their own applications [2] with any programming language that supports sending and receiving HTTP requests such as JavaScript and Python. The advantage of providing WoT service is that reuse of existing HTTP web standards will allow every device to finally be treated as other resources on the web, which makes it much easier to integrate physical devices with any other web content such as Twitter, RSS Feed, etc [4].

There are currently two schools of thought in developing web services: SOAP and REST. The acronym REST stands for Representational State Transfer. RESTful style of web service was a reaction to the more heavy-weight SOAP-based standards. It is the best pattern for web applications. REST focuses on avoiding application state, making sure that important concepts of an application scenario are represented as URI-identified resources [5]. This basically means that each unique URI is a representation of some object. We can get the content of that object using a HTTP GET method. Then we might use a POST, PUT, or DELETE method to modify the object. Another emphasis of REST is on simple point-to-point communication over HTTP using plain old JSON or XML.

This work is supported by Students Involved Innovation Project on Campus Informatization in BUPT, China-Finland Cooperation Project on the Development and Demonstration of Intelligent Design Platform Driven by Living Lab Methodology (2010DFA12780), Architecture and Key Technology Research on Broadband Wireless Campus Innovation Testbed (2010ZX03005-003), and Key Laboratory of Universal Wireless Communications, Ministry of Education.

II. ARCHITECTURE

Due to the light-weight and human readable features of the RESTful web service, we present our prototype of a platform using RESTful API to integrate and share the sensor data.

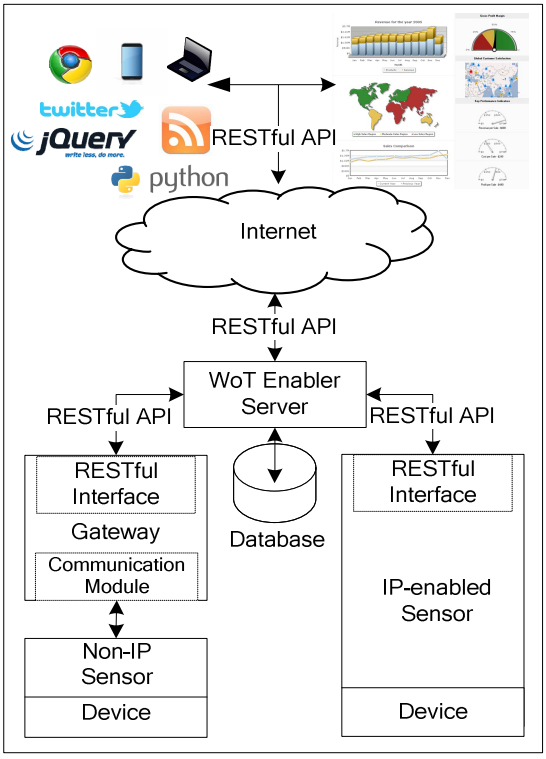


Figure 1. RESTful platform to share sensor data.

As shown in Figure 1, the lowest level includes different kinds of devices, which are connected to sensors such as temperature sensor, humidity sensor, geo-location sensor, etc.

978-1-4244-7255-0/11/$26.00 ©2011 IEEE

Some sensors are IP-enabled, so they are able to be programmed to provide RESTful communication, while others should be connected to gateways, which also have RESTful interfaces. Then the sensors' data are collected by the WoT Enabler Server by making RESTful HTTP requests. The WoT Enabler Server stores the sensors' data to the Database and then opens it to the Internet in the form of RESTful API. Finally, third party developers could implement varies of web applications by making RESTful HTTP request to the WoT Enabler Server and they never need to worry about what the low level interfaces of sensors are.

III. RESTful API

In this chapter we discuss the RESTful API which is designed to help transmit data in a high efficient way between the IP-enabled Sensors/Gateways, the WoT Enabler Server, and some top level Web Applications in our platform.

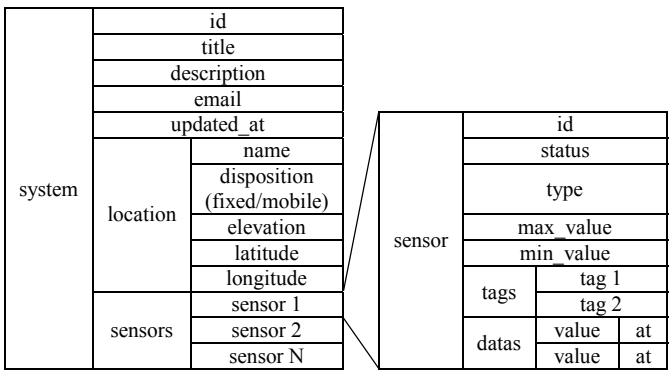
A. Data Structure

We adopt Extended Environments Markup Language (EEML) [6], a protocol for sharing sensor data, to describe the data structure. A breadcrumb-like hierarchical structure is used to represent the sensor data. The structure is "System" > "Sensor" > "Data".

- A "System" is a collection of measured data. The owner of the system defines the system's title and description, as long as its particular geo-location. A system contains all context-specific information of the measured sensors belong to it. The term disposition of a system's location encompasses both fixed entities such as a room, a building or a factory; and mobile entities such as a cell phone or a car, etc.
- A "Sensor" represents an individual sensor within a system. It can specify the data's type, max_value and min_value, and the status to indicate whether the sensor is valid or not, and as well as some user-defined tags for search.
- A "Data" is simply like a key-value pair, which includes a sensor's specific value and the point of the time when this value generates.

Detail of how these labels and values are organized is shown in the following table.

TABLE I. DATA STRUCTURE



In this structure, some properties are always required such as id and title, while others like tags, max_value and min_value are optional [6].

For example, we wire up a research laboratory system with temperature, humidity and CO2 sensors. We create a "System" titled "My Laboratory", with three "Sensors", which could be tagged "temperature", "humidity" and "CO2". Individual "Datas" at a point in time might be "23.2", "34" and "3820" respectively.

This hierarchical structure makes it possible to process data in different levels separately. Going back to our previous example, if we want to draw a chart to illustrate the humidity sensor exclusively, we only need to get the humidity data from the WoT Enabler Server. While if we would like to evaluate the entire system of the laboratory environment, we could get all required data from the Enabler Server by making only one request.

As we mentioned previously, RESTful access uses the following four HTTP verbs to determine which action to make on a particular object [3]:

- GET : Retrieves the current state of the object
- PUT : Sets the current state of the object
- POST : Creates a new object
- DELETE : Deletes the object

In practice, we need an additional method to get all the objects in the same level. This is especially useful for searching and indexing. We call this method LIST. Detail of how and where to make these five requests to is shown in the following table.

TABLE II. RESTful URIs

Method	System	Sensor	Data
LIST	/systems	/systems /<sys_id> /sensors	/systems /<sys_id> /sensors /<sensor_id> /datas
POST			
GET	/systems /<sys_id>	/systems /<sys_id> /sensors /<sensor_id>	/systems /<sys_id> /sensors /<sensor_id> /datas /<timestamp>
PUT			
DELETE			

Unfortunately, some HTTP clients such as normal browsers do not provide native support for making PUT or DELETE requests. In order to work around this limitation, it is possible to send a POST request by adding a method override parameter "_method" at the end of the URL. For example:

- POST to */systems/3.json?_method=put* will simulate a PUT request

- POST to */systems/3.json?_method=delete* will simulate a DELETE request

In addition, several parameters can be applied to limit or refine the returned data. For example, if there is a GET request to */systems.json?tag=CO2&location_name=ChaoyangPark*, it means that the results will only list systems tagged with CO2 and with the location name of ChaoyangPark.

B. Data Formats

Our RESTful API supports three kinds of data formats: JSON, XML and CSV. Each of these formats is best suited to a particular purpose. There are two ways to specify which format we are using:

- Append the format identifier to the URL. For example: */systems.json*
- Pass the "Accepts" header in the HTTP request. For example: *Accept: application/json*

1) JSON Format: The JSON data format [7] can be easily parsed using JavaScript in the browser, so it is particularly suitable for developing web based applications. Additionally, compared to XML, JSON format has much lower processing overheads and uses less bandwidth to transmit [8].

2) XML Format: The XML format contains the same information as JSON. However, it performs better in systems such as building management [9], for many existing systems are still using traditional XML format to transmit.

3) CSV Format: The CSV format doesn't contain metadata as JSON or XML format have, but it is not much of an issue. Its idea is to keep the data as raw as possible. This format is especially designed for use by very simple or low powered embedded devices [10]. A full representation of CSV is as follows:

45,23,2010-08-22 11:23:14,22.8
45,23,2010-08-22 12:23:14,23.2
<system_id>,<sensor_id>,<timestamp>,<value>

IV. IMPLEMENTATION

This part introduces a demo implementation to better describe our RESTful WoT API.

Firstly we built the WoT Enabler Server, which is programmed in the language and framework of Ruby on Rails. We choose Ruby on Rails because it has the highest efficient in deploying a RESTful application. This Enabler Server serves as a core component which handles RESTful HTTP requests and generates RESTful HTTP responses.

Then we pushed the Beijing Air Quality Index (AQI) [11] data, which came from a MetOne BAM 1020 particle monitor located in NE Chaoyang District, to the WoT Enabler Server hour by hour, by means of the RESTful API described in the former chapter. After that we created some tiny applications based on the data provided by the WoT Enabler Server, with the help of RESTful API as well.

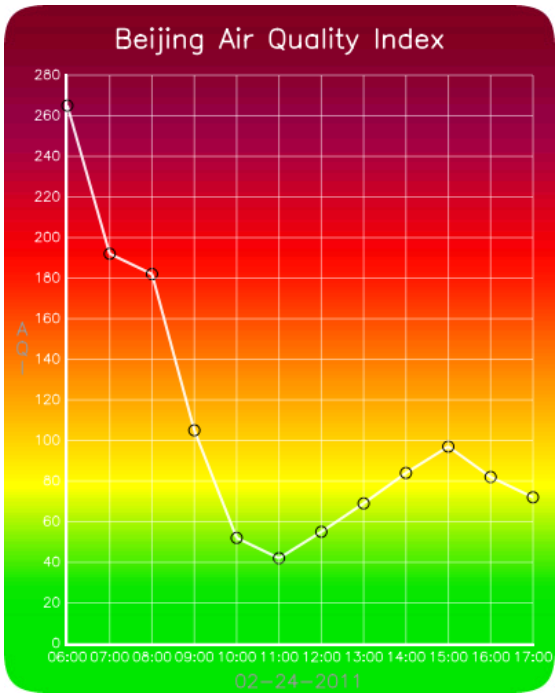


Figure 2. Demo Application: Beijing AQI Monitor.

In the demo application showed in Figure 2. The values of AQI in the latest 12 hours were fetched by making a HTTP GET request to the system URI. The procedures were as follows.

- REQUEST (from Application to WoT Enabler Server)

  GET */systems/32.json?number=12*

- RESPONSE (from WoT Enabler Server to Application)

  HTTP/1.1 200 OK


```
{
"id":32,
"title":"Beijing AQI",
"updated":"2011-12-24 17:00:00 CST",
"location":
{
  "name":"Chaoyang District",
  "lat":39.954914,
  "lon":116.466204,
},
"sensors":[{
  "id":"1",
  "datas":[
    {"at":"2011-02-24 17:00:00 CST ","value":"72"},
    {"at":"2011-02-24 16:00:00 CST ","value":"82"},
    {"at":"2011-02-24 15:00:00 CST ","value":"97"},
    {"at":"2011-02-24 14:00:00 CST ","value":"84"},
    {"at":"2011-02-24 13:00:00 CST ","value":"69"},
    {"at":"2011-02-24 12:00:00 CST ","value":"55"},
    {"at":"2011-02-24 11:00:00 CST ","value":"42"},
    {"at":"2011-02-24 10:00:00 CST ","value":"52"},
    {"at":"2011-02-24 09:00:00 CST ","value":"105"},
```

```
        {"at":"2011-02-24 08:00:00 CST ","value":"182"},
        {"at":"2011-02-24 07:00:00 CST ","value":"192"},
        {"at":"2011-02-24 06:00:00 CST ","value":"265"},
    ],
    }]
}
```

When the Application received the response from the WoT Enabler Server, it parsed the JSON string and displayed the values by some bar charts.

In the other case, when a new piece of data was generated from the sensor, the gateway pushed this data to the WoT Enabler Server by making a create data request. The procedures were as follows.

- REQUEST (from Gateway to WoT Enabler Server)

  POST */systems/32/sensors/1/datas*

  {"at":"2011-02-24 18:00:00 CST ","value":"112"}

- RESPONSE (from WoT Enabler Server to Gateway)

  HTTP/1.1 201 Created

After that, refreshing the demo application would generate new RESTful GET request to the WoT Enabler Server, and the response will certainly result in new AQI values to be shown.

All these processes of RESTful requests and responses are human readable and light-weighted. We only need to focus on the URI and the four HTTP verbs of each single object or resource.

V. CONCLUSION AND FUTURE WORK

This paper introduces a platform to share sensor data using RESTful WoT API. It points out the data structure, data formats and web URIs of the API. The Beijing AQI demo application indicates that the API is helpful in building varied applications.

However, the reported work also exposes a number of issues that need further investigation. Centralized computing and data storage of this platform brings enormous pressure to the Core Web Server. Cloud computing and storage may be a good solution to solve this concern. Privacy is also an important aspect that should be considered to avoid data abuse. Therefore user authentication and appropriate access control strategy should be introduced to the RESTful WoT API as well, making the RESTful API helpful in sharing sensor data almost as easy as browsing the web.

[1] A. Dunkels, J. P. Vasseur, “IP for Smart Objects,” Internet Protocol for Smart Objects (IPSO) Alliance White Paper No.2, September 2008.

[2] V. Trifa, D. Guinard, “Towards the Web of Things,” Whitepaper 1.0, 2010.

[3] R. T. Fielding, “Architectural Styles and the Design of Network-based Software Architectures,” Ph.D. dissertation, University of California, Irvine, 2000.

[4] D. Guinard, V. Trifa, “Towards the Web of Things: Web Mashups for Embedded Devices,” Sixth International Conference on Networked Sensing Systems (INSS), 2009.

[5] E. Wilde, “Putting things to REST,” Technical Report UCB iSchool Report 2007-015, School of Information, UC Berkeley, November 2007.

[6] Extended Environments Markup Language, EEML, http://www.eeml.org/.

[7] JavaScript Object Notation, JSON, http://www.json.org/.

[8] D. Yazar, A. Dunkels, “Efficient Application Integration in IP-Based Sensor Networks”, Swedish Insitute of Computer Science, November 2009.

[9] J. Polley, D. Blazakis, J. McGee, D. Rusk, J. S. Baras, “ATEMU: A Fine-grained Sensor Network Simulator,” First IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON’04), 2004.

[10] T. Roscoe, L. Peterson, S. Karlin, M. Wawrzoniak, “A Simple Common Sensor Interface for PlanetLab,” PlanetLab Design Notes PDN-03-010.

[11] Twitter Feed of BeijingAir. U.S. Embassy. https://twitter.com/#!/beijingair.