A Web Platform for Globally Interconnected 6LoWPANs

Zengxu Yang, Xiaofei Guo, XiaoZheng Guo, David J. Janowsky, C. Hwa Chang
The Tufts Wireless Lab
Department of Electrical and Computer Engineering
Tufts University
zengxu.yang@tufts.edu, xiaofei.guo@tufts.edu, willguo@mathworks.com,
david.janowsky@tufts.edu, chorng.chang@tufts.edu

Abstract

The Internet of Things (IoT) enables a variety of application scenarios with daily life objects, some of them equipped with sensors or actuators, connected to the Internet. The standardization of the Constrained Application Protocol (CoAP) on the resource-constrained devices by Internet Engineering Task Force (IETF) facilitates the integration of low-power wireless networks into the Internet. A full-stack Web application is designed to communicate with the low-power constrained nodes over globally deployed IPv6 over Low Power Wireless Personal Area Networks (6LoWPANs) through CoAP. This Web application is based on Java Spring MVC framework and can be divided into three parts: communicating with the sensors, deploying the database and visualizing the data.

1 Introduction

As more and more smart devices such as wireless personal devices, smart objects, embedded systems and remote sensors become prevalent, the integration of such devices into the Web is more and more important so that end users can easily access their devices over the Internet. The network address space of IPv4 is running out and thus has to be expanded to support the Internet services of massive numbers of pervasive devices for various uses[5, 3]. Internet Protocol version 6 (IPv6) was hence formed. 6LoWPAN makes IP usable in many low-power and lossy wireless networks[7]. An overview of 6LoWPAN and its implementations is describe in literatures[11]. These networks are enabling a completely new and never imagined generation of devices, applications and services where IoT resources can be conceived as service end-points, i.e. Things or Resources as a Service (TaaS or RaaS)[2]. For example, manufacturing using Wi-Fi, Message Queuing Telemetry Transport(MQTT) and 6LoWPAN has been implemented[12].

In this aspect, the new protocol to support the communication between Web clients and the constrained devices through the server is becoming important. For these networks, User Datagram Protocol (UDP) rather than Transmission Control Protocol (TCP) is more commonly used due to its short real-time delay. Based on UDP, CoAP is proposed by IETF to be used on IoT[6]. CoAP is a specialized Web transfer protocol for use with constrained nodes and constrained (e.g., low-power, lossy) networks. The nodes often have 8-bit microcontrollers with small amounts of ROM and RAM, while constrained networks such as 6LoWPANs often have high packet error rates and a typical throughput of 10s of kbit/s. The protocol is designed for machine-to-machine (M2M) applications such as smart energy and building automation. CoAP provides a request/response interaction model between application endpoints, supports built-in discovery of services and resources, and includes key concepts of the Web such as URIs and Internet media types. CoAP is designed to easily interface with HTTP for integration with the Web while meeting specialized requirements such as multicast support, very low overhead, and simplicity for constrained environments[8]. Much research has been conducted on the architectures of the networks and the performance of the CoAP [5]. We deployed an IoT system consists of two platforms: an 6LoWPAN platform[10] and a Web platform to connects Web clients over HTTP and 6LoWPAN IoT devices over CoAP. The IoT system can be modeled as shown in Figure 1. In our IoT system, the module (a) represents the 6LoWPANs with four constrained sensor nodes in the 6LoWPAN platform. As mentioned earlier, this platform uses CoAP. The module (b) is a translating module, this module is the communication block that translates between CoAP and HTTP.. The module (c) is the main back-end block. Inside this block, some user requests that interact with the constrained devices and database will be processed. Also, some back-end services that are not directly involved with user interactions, such as collecting and processing the data, locking the database, etc., will be provided through this block. The module (d) represents the database, where the platform stores the data the constrained devices produced. The module (e) is the internet server that handles users' requests, provides the request information, like accessing the database, configuring the platform, and controling the constrained nodes, for the back-end platform. This article focuses on the design and implementation of mod-

International Conference on Embedded Wireless
Systems and Networks (EWSN) 2019
25–27 February, Beijing, China

ISBN: 978-0-9949886-3-8

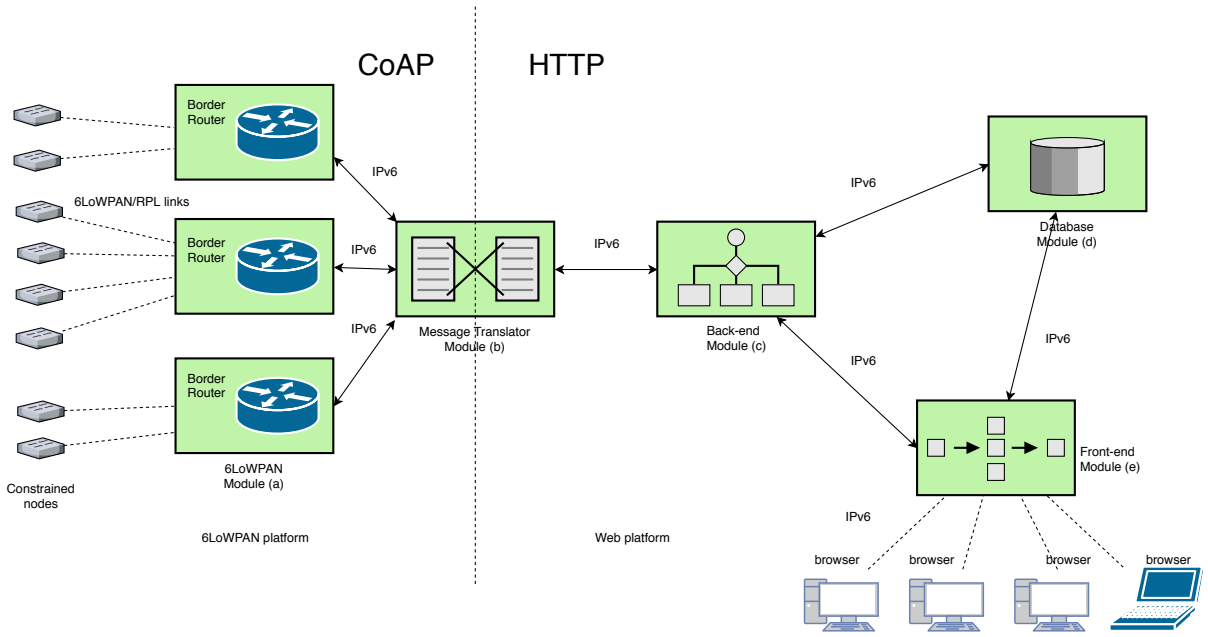

Figure 1. IoT system block diagram

ule(b), (c), (d) and (e). The implementation is mainly based on Java, with limited usage of other front-end languages. Java became the predominant platform for mission-critical enterprise applications after its initial launch in 1997. Several common services for enterprise application became standardized as J2EE (Java 2 Enterprise Edition) later called Java EE. The Spring Framework provides a comprehensive programming and configuration model for modern Java-based enterprise applications. It is a proven, stable, high-quality platform ubiquitous in today's Enterprise Java world and can be integrated into a Java EE environment. With benefits like Dependency Injection, Aspect Oriented Programming, and Enterprise Service Abstraction, numerous large companies have deployed Spring in mission-critical applications with very good results[9].

2 Design and Implementation

2.1 Overview

The main idea in this article is to build the Web platform as a set of stateless reusable REST services. The Representational State Transfer (REST) style is an abstraction of the architectural elements within a distributed hypermedia system. REST ignores the details of component implementation and protocol syntax in order to focus on the roles of components, the constraints upon their interaction with other components, and their interpretation of significant data elements. It encompasses the fundamental constraints upon components, connectors, and data that define the basis of the Web architecture, and thus the essence of its behavior as a network-based application.The fundamental concept of REST is that anything is a resource that can be accessed or manipulated[4]. These states need to be represented using a common format such as XML or JSON. In the case of Web applications, HTTP is commonly used to support a RESTful architecture. In other words, REST is used to create a Web application that can be accessed by an HTTP API. Standard HTTP methods such as GET, POST, PUT, and DELETE are used to access and manipulate REST Web resources. The CRUD operations have four basic persistent functions: create, read, update, and delete. The Web platform consists of the front-end (HTML, CSS, JavaScript/jQuery), the back-end (Spring MVC, logical layer, translating module), and the database (Hibernate, MySQL). The communication between the client side and the server side happens through a RESTful API. The front-end pages involve everything that the user sees, including interaction with the users, as well as displaying data in a well-defined style. The technological stack of the front-end part for the Web platform is HTML, CSS, JavaScript, BootStrap, and Java Server Page (JSP). BootStrap is a JavaScript library and JSP is used for generating dynamic contents. The back-end is built with Java which connects with the database and the 6LoWPANs to save or update data or settings of constrained nodes and return the result to the end user in the form of front-end code such as HTML elements or contents. The back-end is implemented with Spring Framework, which is popular for building the back-end features in enterprise applications. It has a configuration module where Spring handles many common concerns such as handling HTTP requests, connecting to databases, and it allows the developer to focus on business services. The developer develops business services and annotates classes with Spring-provided annotations, which lets Spring know about those services. It also provides infrastructure to access the database, such as Hibernate, and map the Java class into the databases entries. The database is built with MySQL, which is the leading open-source database solution for many well-known application such as Facebook, Twitter, YouTube, etc.[1].

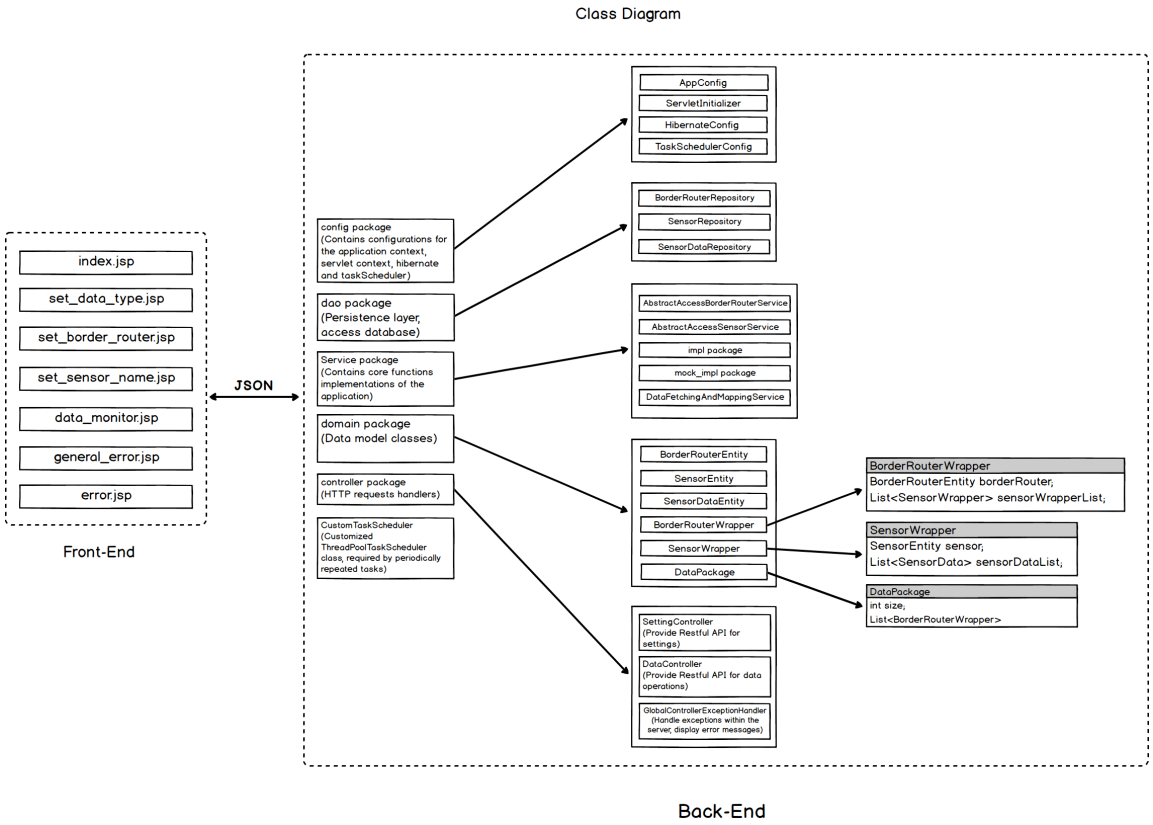

Figure 2. Class diagram of the application

2.2 Design

The functionalities of the Web platform are highly driven by the user experience, and below are some user cases we start with:

1. The communication between the application and the sensor network using a variety of protocols.
2. Users can access our application through a GUI interface in their browser.
3. Users can add border routers by entering border router IP addresses and border router names in the GUI interface.
4. The server can automatically explore the sensor network to add sensor nodes connected to each border router.
5. Users can set data types that sensors will report to the server.
6. Users can monitor data from each sensor through the GUI interface.
7. Users can toggle auto data fetching feature and see the trend of data over time.
8. Users have two options when they open our application. They can either start a fresh application or start from what they saved from last time.

The application can be divided into three parts, front-end, back-end and database.

Front-end: The front-end module is the interface for the end-users with graphic options in the browser. For technology stack, this part contains all `.jsp` pages, which is developed with BootStrap, JavaScript and JSP. The front-end part provides a visual interface for users to configure the system and monitor data from sensors. These `.jsp` files are not directly visible to public, therefore they need `jspViewResolver` and `PageController` to map specific URL patterns to them.

Back-end: The back-end module contains all the core logic of the project. It gives front-end working functionality including the communication with the sensor network, storing and fetching data from the database and the management of front-end page flow. In this project, we chose Java back-end tech stack (Java, Spring MVC, Hibernate) to build the back-end part.

Database: The database is built with MySQL and can be integrated with the back-end Spring Framework through Hibernate.

Figure 2 shows the overall class structure of the application.

2.3 Implementation

The application has been implemented according to the latest Web 2.0 standards. The main welcome page is under `webapp/resources` with the name `index.jsp`. In this page, users can be navigated to the information page or the service page. Once the service page is activated, the user is required

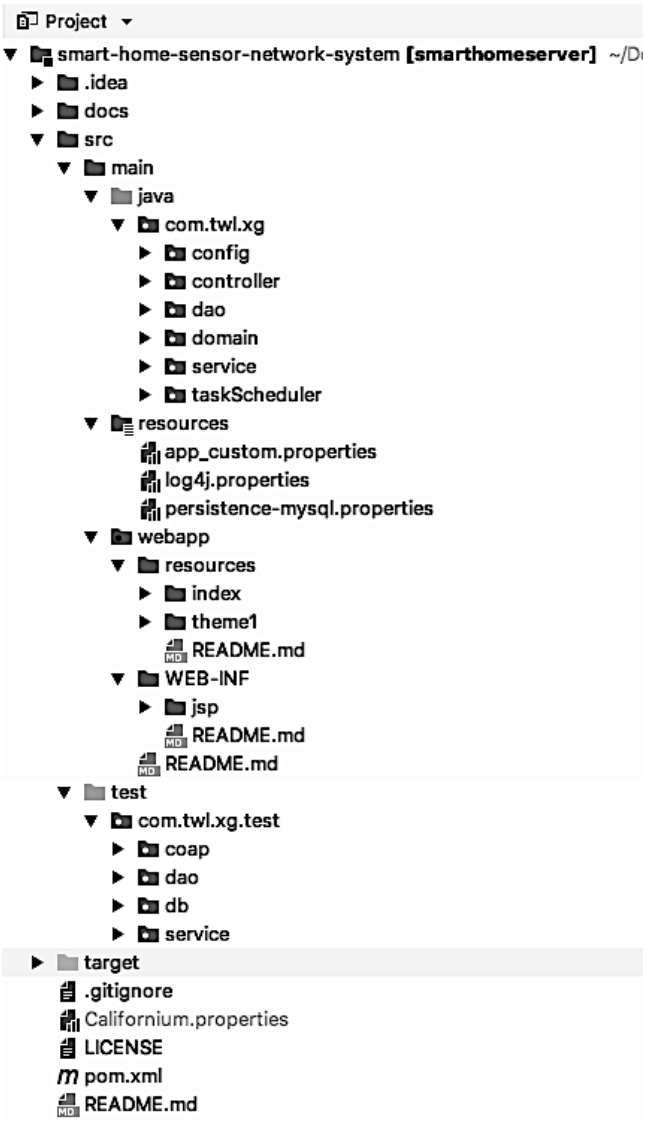

Figure 3. File structure of the application

to set up the environment parameters like border router address, sensor address, data type, etc. The setting page is comprised of technologies including Bootstrap, HTML, CSS and JavaScript. Once the setting has been done, the session begins, and the user is redirected to the `monitor_data` page where they can view the charts of data from their queries. In the back-end, the application queries the data according to the parameters the HTTP message has passed in from the service setting page. If the data is already in the database, all of it will be passed to the front-end templates powered by JSP. The display of data in the browser will be then powered by AJAX.

Figure 3 shows the file structure.

- `src`: Contains all the source code and resources of the project
  - `src/main`: Contains all the source code and resources except for test code
  - `src/main/resources`: Contains resources needed by the back-end part, in this case this folder stores all the Spring `.properties` configuration files;
  - `src/main/webapp`: This is the Web application resource folder and it contains static resources and .jsp files of the front-end part;
    - `/src/main/webapp/resources`: Contains all the static resources including `.js`, `.css` files and images;
    - `/src/main/webapp/WEB-INF`: Contains all things that aren't in the document root of the application. These files cannot be served directly to a client by the container, but they are visible to servlet code;
      - `/src/main/webapp/WEN-INF/jsp`: Contains all `.jsp` files which implement all of the front-end interfaces;
  - `src/test`: Contains all the unit test code
- `pom.xml`: This POM file contains information about the project and configuration details used by Maven to build the project;

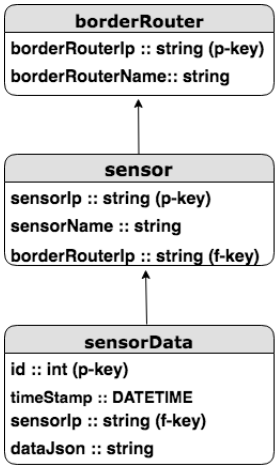

Figure 4. ER diagram of our database schema

MySQL database is used to store border router and sensor information and data sets. Figure 4 is the ER diagram of our design.

Three tables are contained in the database: `borderRouter`, `sensor` and `sensorData`. The `borderRouter` and `sensor` tables are applied a one-to-many relationship. `sensor` and `sensorData` tables also have a one-to-many relationship.

Index Page:

This page is the entrance of the whole system. To start, you need to click the "GET STARTED" or the "CONTINUE" button. If it is your first time using our system, you have to click "GET STARTED". Figure 5 shows our index page.

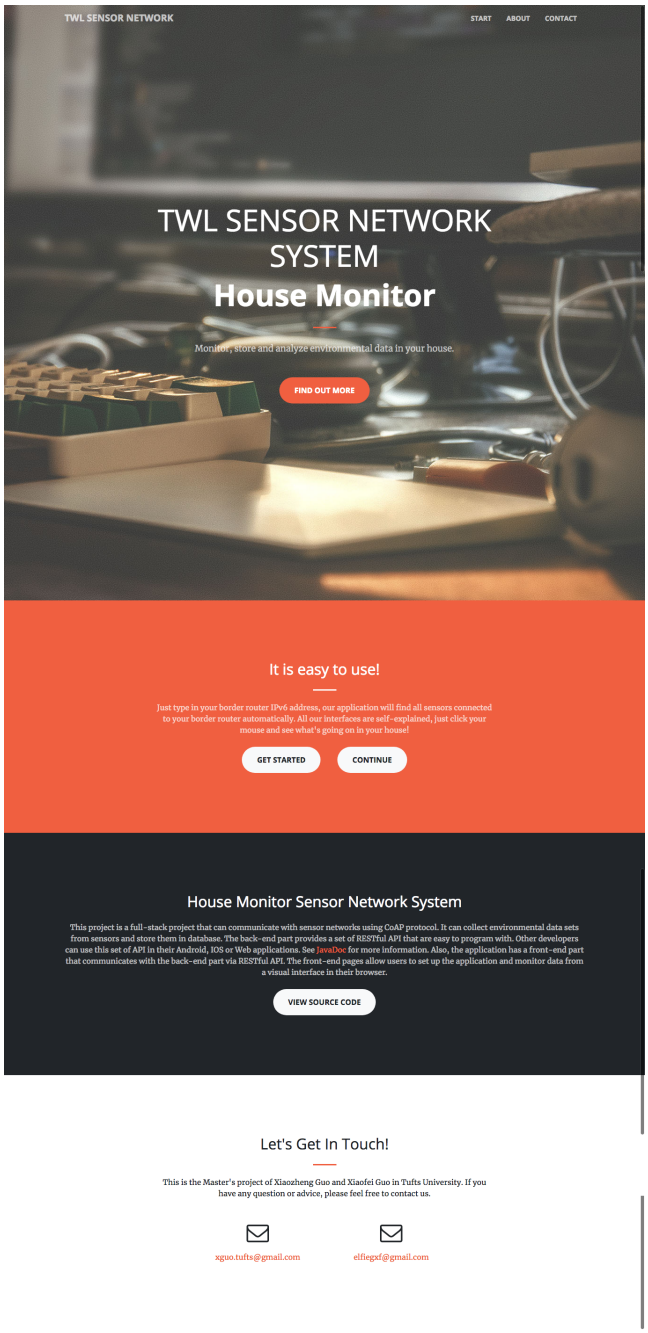

Figure 5. Index page

Data Type Setting Page:

After clicking "GET STARTED", you will be redirected to this page. In this page you need to enter the types of data that you want from sensors. Follow the instructions on the page, then click "Submit" or "Skip" after you finish. The Figure 6 shows the sample setting page. Other setting pages are alike.

Border Router Setting Page:

The next step is to enter the IP addresses of the border routers in your sensor network. It is required for you to set a name for each border router. Follow the instructions on the page and click "Submit" after you finish.

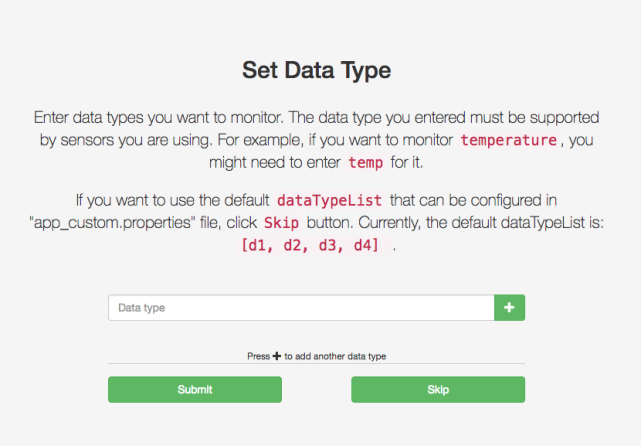

Figure 6. Data type setting page

Sensor Name Setting Page:

After you entered border router IP addresses, our server will automatically find all sensors that are connected to each border router. The next step is to set a name for each sensor. This step is optional, follow the instructions on the page and click "Submit" or "Skip" after you finish.

Data Monitor Page:

On this page, two forms of data are displayed. One is the current data displayed in bar charts, the other one is the history data that is displayed in line charts. Click "Show Current Data" and "Show History Data" in the top bar to switch between these two modes. The left side bar contains a list of border router names, and clicking one of them selects the sensor data you want to see. The button "Auto Data Fetching Disabled/Enabled" toggles the auto data fetching function. Before checking history data, you should enable auto data fetching to store data into the database. The last button is "Clear History Data", click this button to clear all sensor data entries in database.

3 Results

This section displays the results after running the application. Figure 7 shows the data monitor page with a bar chart of current humidity, lightness data of the all the sensors belonging to the selected border routers. Figure 8 shows the curve chart of the history data.

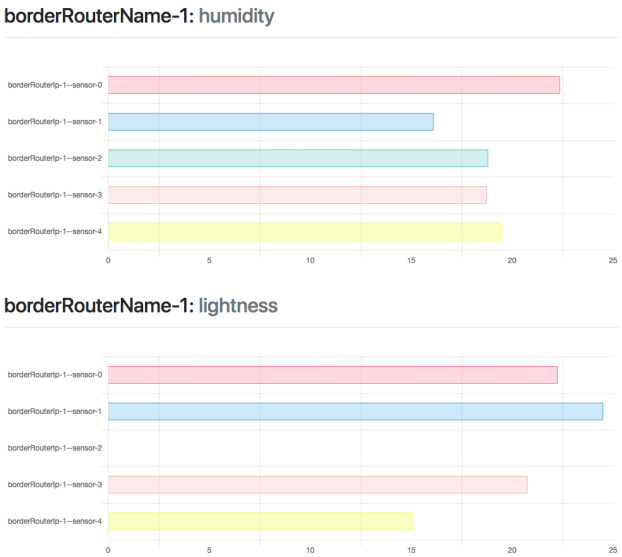

Figure 7. Current data bar chart page

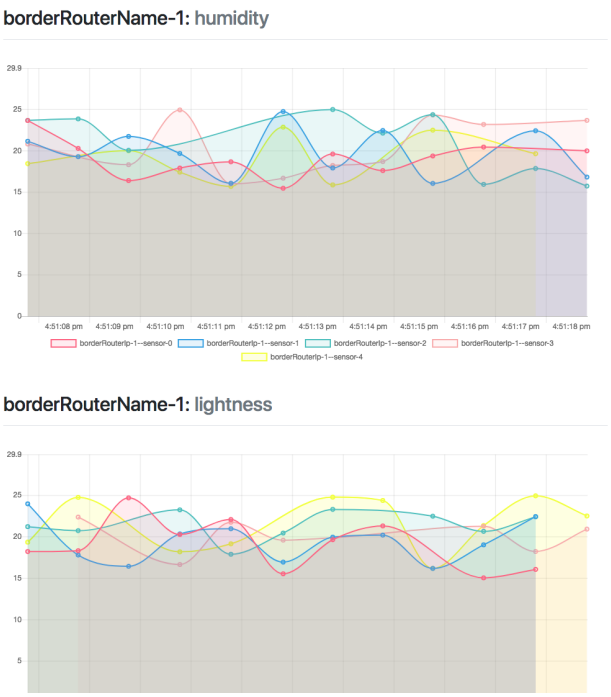

Figure 8. History data line chart

Figure 7 lists the current humidity and lightness data of five different border routers in the data bars. The value of bars can be found on the x-coordinate. From this figure, the comparisons between different border routers can be clearly shown. Figure 8 shows the history data, the x-coordinate shows the timestamp, and the y-coordinate shows the value of data. Users can easily see the trends.

4 Conclusion

The Tufts Wireless Lab is developing a Web platform for globally interconnected 6LoWPANs, which is made up of four modules: the back-end module, the front-end module, the database module, the message translator module. This Web platform can be connected to the 6LoWPAN platform over CoAP[10]. The first three modules have been fully implemented and well functioning. The implementation allows the users to set up the environment according to their specific constrained devices and also provides the interface for users to query the data that they want to visualize. The front-end module was built with BootStrap, JavaScript and JSP. The back-end module was implemented with Java back-end tech stack: Java, Spring MVC, Hibernate. The database module used MySQL. The message translator module is under development and will be integrated into the platform in the near future.

5 References

[1] Top 10 Reasons to Choose MySQL for Next Generation Web Applications. Technical report, Oracle Coporation, 2016.

[2] M. Castro, A. J. Jara, and A. F. Skarmeta. Enabling end-to-end CoAP-based communications for the Web of Things. *Journal of Network and Computer Applications*, 59:230–236, Jan. 2016.

[3] M. Dooley and T. Rooney. IPv6 DEPLOYMENT AND MANAGEMENT. page 29.

[4] R. T. Fielding. in Information and Computer Science. page 180, 2000.

[5] C. Gomez, J. Paradells, C. Bormann, and J. Crowcroft. From 6lowpan to 6lo: Expanding the Universe of IPv6-Supported Technologies for the Internet of Things. *IEEE Communications Magazine*, 55(12):148–155, Dec. 2017.

[6] M. R. Palattella, N. Accettura, X. Vilajosana, T. Watteyne, L. A. Grieco, G. Boggia, and M. Dohler. Standardized Protocol Stack for the Internet of (Important) Things. *IEEE Communications Surveys & Tutorials*, 15(3):1389–1406, 2013.

[7] S. Raza, H. Shafagh, K. Hewage, R. Hummen, and T. Voigt. Lithe: Lightweight Secure CoAP for the Internet of Things. *IEEE Sensors Journal*, 13(10):3711–3720, Oct. 2013.

[8] Z. Shelby, K. Hartke, and C. Bormann. The Constrained Application Protocol (CoAP). Technical Report RFC7252, RFC Editor, June 2014.

[9] E. Wolff. Spring - A Manager's Overview. page 15, 2008.

[10] Z. Yang and C. H. Chang. A 6lowpan IoT Platform on the Global Internet. 2019. To be published.

[11] Z. Yang and C. H. Chang. 6lowpan Overview and Implementations. 2019. To be published.

[12] T. Zeybek and C. H. Chang. An IoT Implementation for Manufacturing using Wi-Fi, 6lowpan, and MQTT. 2019. To be published.