ENVIRONMENTAL SENSOR MONITORING WITH SECURE RESTFUL WEB SERVICE

Sungchul Lee¹, Ju-Yeon Jo², Yoohwan Kim³
Department of Computer Science
University of Nevada Las Vegas, Las Vegas, NV 89154
², ³ Email: {Juyeon.Jo, Yoohwan.Kim}@unlv.edu
¹ Email: lees173@unlv.nevada.edu

Abstract

The Nevada Solar Energy-Water-Environment Nexus project generates a large amount of environmental monitoring data from variety of sensors. This data is valuable for all related research areas, such as soil, atmosphere, biology, and ecology. An important aspect of this project is promoting data sharing and analysis using a common platform. To support this effort, we developed a comprehensive architecture that can efficiently collect the data from various sensors, store them in a database, and offer an intuitive user interface for data retrieval. We employed Arduino-based sensors due to their flexibility and cost-effectiveness. Restful Web Service is used for communication with the Arduino-based sensors, and Google Charts service has been used for data visualization. This framework for sensor data monitoring with Web Service is expected to allow the Nevada Nexus project to seamlessly integrate all types of sensor data and to provide a common platform for researchers to easily share the data.

Keywords: Web Service, Arduino, Restful, Sensor Network, ID-Based Encryption

1. INTRODUCTION

1.1 Need For Environment Monitoring

Developing renewable energy resources is a national priority (U.S. Office of Management and Budget, 2012). In order to develop and manage the renewable energy resources efficiently, University of Nevada, Las Vegas (UNLV) and other Nevada institutions are collaboratively working on the Nevada Energy-Water-Environment Nexus project. This project explores advanced knowledge through the research on solar energy generation, its environmental impact, and associated water issues. To process and share the collected data, it is building a new capability in cyberinfrastructure (CI) (Nevada Solar Nexus Proposal, 2013). To this date, Nexus project has generated a large amount of data from investigating the environmental impacts and water issues of solar energy production by engineers, hydrologists, biologists, ecologists, soil scientists, atmospheric scientists, and economists (Nevada Solar Nexus Proposal, 2013). Standardized and integrated data on solar, water, and environment are needed to handle the needs of their research. The project also requires real-time data collection and distribution to variety of entities to support their research.

Over the years, environmental monitoring data has been stored and analyzed at the field devices (Zheng, Tao., Qin, Yajuan., & Zhang, Hongke, 2010), and the collected data has been moved to database manually (Nikos, Giannopoulos., Christos, Goumopoulos., & Achilles Kameas, 2009).

In this paper, we employ Web Service with which the data can be directly stored at database and analyzed conveniently. The security features, i.e., confidentiality, integrity and availability, are also better supported with the new data sharing system based on the Web Service.

1.2 Network Infrastructure Status

![NEW-STAR facility](Figure 1. NEW-STAR facility)
In order to achieve the goals mentioned above, new facility and network infrastructure is needed. There have been preceding projects in Nevada with similar goals before the current Nexus project. The existing facilities from those projects are now being leveraged. The NEW-STAR facility, standing for “Nevada Environment, Water, and Solar Testing and Research”, was built in 2008 and being used in the current project. Figure 1 shows the property and the solar thermal power plant adjacent to it. The NEW-STAR facility is connected to other research locations throughout Nevada via a network infrastructure that is illustrated in Figure 2. One of the needed improvements for the cyberinfrastructure in the Nexus project is to increase the connectivity and to expand the existing network infrastructure. The infrastructure improvement plans include expanding the scope of the existing Sensor and Nevada Climate Change Portal (NCCP) system into the new Nevada Research Data Center (NRDC) along with an extended sensor network, comprising additional communication links, servers, and databases (The Solar Energy-Water-Environment Nexus in Nevada, 2014). NCCP site was built to accommodate Energy-Water-Environment Nexus project. It has been collecting data and constructing data publications since 2011. NCCP site established the Nevada climate-eco-hydrology assessment network (NevCAN) for collecting environmental sensor data such as, precipitation, pant canopy interception of snow, subsurface soil water flow, soil water content, snow depth, soil temperature, thermal flux, solar radiation, and so on (Nevada Climate Change Portal, 2014). NCCP site can store environmental sensor data over four hundred millions data points per year by NevCAN.

There are a few issues to be addressed in building the new system. Due to limited software support and data usability, the previous system has not reached its full utilization. For example, it is challenging to share the data generated from the monitoring devices with other researchers. From the data management perspective, there are two reasons for this underutilization. First concern is the data integrity, i.e., the collected data must not be changed by any unauthorized person. It hasn’t been much of a problem so far since most researchers conducted their study independently with the original data. However, to make it sharable, the data integrity issue must be resolved because any unauthorized change may make the entire database unusable. Second concern is the data standardization. Researchers are producing their own type of data in an ad-hoc fashion while processing and storing the data. Very little attention is given to interoperability. As a result, these data are not easily searchable or understandable by other researchers or their tools. In order to solve these issues, the concept of Sensor Data as a Service (SDS) has been applied in the Nexus project. SDS has been proven to be suitable for Data-Centric Service in a previous research (Jia, et al., 2013). For its implementation, we decided to make a REST-base Web Service (Jia, et al., 2013). Regarding the data input, we employed Arduino boards (Hiro, et al., 2013) to collect data from the sensors.

The proposed architecture is composed of three main parts, that is, sensor network, RESTful web service and database. These parts are covered in the following sections. Section 2 describes the motivation and the background on REST-based Web Service in the context of NV Nexus project. Section 3 illustrates the process of current sensor data.

Figure 2. Network Infrastructures (Frederick, C., & Harries, Jr, 2010).
collection. The database normalization process is described in Section 4. Section 5 describes how the environmental data is collected on Arduino and how it is transmitted to the server via RESTful communication. Section 6 explains about REST architecture and current authentication system for RESTful web service. Section 7 shows how the RESTful web service authenticates sensors via ID-based crypto. Section 8 illustrates the interaction mechanisms among our extended web service,

Google Charts service, and Arduino boards. Section 9 shows how NCCP data is visualized via Google Chart. And we explain about Google Charts. Finally section 10 summarizes our research and outlines the future works.

2. BACKGROUND AND MOTIVATION

In the predecessor projects of the current Nexus project, the researchers stored significant amount of data on their database for almost ten years. The accumulated data is classified into three groups; (1) solar energy generation, (2) water resources, and (3) environment (The Solar Energy-Water- Environment Nexus in Nevada, 2014). It can be applicable to other researches in various fields, such as engineering, hydrology, biology, ecology, soil, atmospheric, economy, and so on. By sharing and combining the data, a high level of synergy can be achieved among researchers in various fields. In spite of the benefits from the synergy, there are three major issues in sharing the data. First, the data gathered from the Nexus project exhibits different types and formats, making it impractical for researchers to process the data when necessary. Moreover, it has been shown that regenerating the data not only takes long time but also produces numerous errors. As a result, researchers tend to overlook the data from other fields or other researchers.

Second, different types of data make the searching operation inefficient. Furthermore, different researchers employ different ways of storing data. There are practically no search engines currently for researchers to get specific data. Although it offers some hints to examine the header section of data, researchers still have a difficulty to understand the proper usage of data. Therefore, researchers end up spending a substantial amount of time for manually searching for specific data from the massive amount of data.

Last issue is security. Among the 3 aspects of data security, i.e., confidentiality, integrity and availability, data integrity is the greatest concern in our case. Researchers are concerned about data being changed improperly. They are reluctant to expose the data due to the possibility of damage to the data. Also, it is difficult to protect the data in a uniform way as researchers employ different data storage methods such as text file, excel file, Oracle database, MySQL database, etc. To resolve these issues in relation to sharing data, Nexus CI team has been developing a scheme for managing, processing, and protecting data.

The answer to the above requirements was an architecture consisting of a sensor network infrastructure, RESTful web services, and database to integrate real-time data. Each component will be discussed in the following sections.

3. CURRENT SENSOR SYSTEM

Nexus project and its predecessor projects have been collecting various environmental data throughout the Nevada state. Some of the sensors are located at UNLV campus for measuring wind and solar data. The collected data has been stored at a CR1000 unit that is shown in Figure 4. CR1000 is one of the most widely used data logger in the industry (Campbell Scientific, 2014). It offers a broad range of measurements and control functions. Figure 3 and Table I illustrates a sample text output of the original data collected at a CR1000 unit and the characteristics of data each element. It contains the measurements of average solar power, total solar

Table I. Data field Description

<table>
<thead>
<tr>
<th>Record</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slrkw_Avg</td>
<td>Avg solar power</td>
<td>km/m²</td>
</tr>
<tr>
<td>SlrMJ_Tot</td>
<td>Total solar power</td>
<td>MJ/m²</td>
</tr>
<tr>
<td>T109_K_Avg</td>
<td>Avg Inside Temperature</td>
<td>Kelvin</td>
</tr>
<tr>
<td>T110_K_Avg</td>
<td>Avg Outside Temperature</td>
<td>Kelvin</td>
</tr>
<tr>
<td>Avg_Wind_Speed</td>
<td>Avg Wind Power</td>
<td>m/s</td>
</tr>
<tr>
<td>Avg_Wind_Dir</td>
<td>Avg Wind Degree</td>
<td>Deg</td>
</tr>
</tbody>
</table>

Figure 3. Sample Data from CR1000

Figure 4. CR1000 in UNLV
power, average inside temperature, average outside temperature, average wind power, and the wind direction in degree. Researchers have to convert them to Excel or CVS format in order to process them in MATLAB. Also, when a specific data entry should be analyzed at MATLAB, researchers have to manually search through the entire dataset to locate the desired data item. In order to change certain functions of CR1000, such as collected data type, measurement unit, or time resolution, it is necessary for the researchers to this manual process is inconvenient and time-consuming, which lowers the productivity of the researchers.

To mitigate the inconvenience, we created a web-based control interface. The sensor data collection functions can be controlled via web services. Web Service also bridges the data with the database and allows improved search and analysis operations.

4. DATABASE CONSTRUCTION

We explain the process of database construction as well as the normalization process briefly in this section. In order to store the raw data into the database, normalization is necessary, as the data from CR1000 is not normalized. A normal form of data works as a restriction on the database and prevents certain undesirable properties from the database (Abraham, Henry, Korth, & Sudarshan, 2011).

In order to normalize the data, the data types and their relationship need to be identified first. There are several entities that will be used in database, such as collector, timestamp, sensor, value, and station. One sensor should be in one collector. The value is defined by one sensor with one timestamp. The using the relationship among the entities in CR1000, we derive the mapping in figure 5.

In order to generate normalized data, we must go through a four-step process (Amir, Mahmoud, & Behnam, 2008). The four-step process is:

1. 1NF: atomic no list, no set
2. 2NF: fully dependent no partially dependent
3. 3NF: transitive dependent x \rightarrow y, y \rightarrow z, y \rightarrow a
4. BCNF: only key can decision. x \rightarrow y, x must be key

1NF represents a relation scheme, expressed only in atomic form, where the elements of the domain are considered indivisible units. The values in the domain may not represent a list, set, or any other composite values. 2NF is a relation scheme where every non-prime attribute fully depends on every key. 3NF is a relation scheme where no non-prime attribute is transitivity-dependent upon a key. Boyce-Codd Normal Form (BCNF) is a relation scheme where every determinant is a candidate key. Figure 6 displays the relations and structures among the tables in the database. Sensor table shows which sensors can be included in a particular collector type (Collector_ID). Installed Collector table shows the location of a collector represented by Collector_ID. Station table describes the location, such as address, zip code, and so on. Measure table shows the value of data at a Timestamp in a Sensor_ID of a Collector_ID. By transforming the data into a normalized form, SQL join method can be performed without causing errors. Furthermore, different fields of data can be combined safely. We construct database on MariaDB (COMPUTERWORLD, 2014) based on the relations defined in Figure 6.

5. ARDUINO-BASED SENSOR DATA COLLECTION

To overcome the limitations of CR1000, we explored the possibilities of data collection using Arduino (Campbell Scientific, 2014). Arduino is an open-source electronics prototyping platform that focuses on flexible, convenient hardware and software. Arduino can create interactive objects and environments for beginners, such as, artists, designers, hobbyists, and so on (Arduino, 2013).

Table II. CR1000 Vs. Arduino Specifications

<table>
<thead>
<tr>
<th></th>
<th>CR1000</th>
<th>Arduino</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microcontroller</td>
<td>16 bit H8S Renesas</td>
<td>ATmega328</td>
</tr>
<tr>
<td>Input voltage</td>
<td>-5 V ~ 5V</td>
<td>7~12 V</td>
</tr>
<tr>
<td>Analog pin</td>
<td>16</td>
<td>6</td>
</tr>
<tr>
<td>Digital I/O pin</td>
<td>8</td>
<td>14</td>
</tr>
<tr>
<td>Flash memory</td>
<td>2MB</td>
<td>32K</td>
</tr>
<tr>
<td>Data storage</td>
<td>4MB</td>
<td>External</td>
</tr>
<tr>
<td>Memory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Price</td>
<td>$1280</td>
<td>$25</td>
</tr>
</tbody>
</table>
The impact of a theft is much smaller. The workload of the web segment that transforms the voltage to temperature, getData( ) every second. The next part is the main operation in a loop form. Within the loop, the timer variable t keeps reading the data using getVoltage ( ) and then sends the temperature data to the server.  

![Figure 7. Connection between Arduino and Sensor](http://hipore.com/ijsc)

Table II shows the characteristics of CR1000 and Arduino. CR1000 has a higher performance microcontroller and larger memory than Arduino. In many cases, sensors are in outside without protection from theft or damage. Arduino is much cheaper than CR1000, thus the impact of a theft is much smaller. Arduino-based sensors can be reliably monitored and controlled remotely. Arduino can also work as a web server making it easier to communicate using standard HTTP communication API. As stated in Table II, Arduino is capable of handling a comparable amount of data to CR1000 (20 vs. 24 inputs). Arduino boards can be equipped with a variety of sensors and function as a datalogger (Vikatos, et al., 2011). They can configure the sensors, such as measurement time resolution, and control switches or actuators. Arduino programming language (based on Wiring) and the Arduino development environment is used for programming the on-board microcontroller. Arduino programs can run either in stand-alone mode or be controlled by the software running on a host computer (e.g. Flash, Processing, MaxMSP). Figure 7 shows an Arduino Uno 3.0 board with an Ethernet Shield V2.0 and a temperature sensor along with their connections. Arduino checks the voltage from the temperature sensor using 5V input voltage. Arduino Ethernet Shield V2.0 is connected to the Internet via RJ45 port (Arduino, 2013). Arduino uses a polling method to measure the voltage from the sensor. The polling interval can be determined by setting a timer in the program. To minimize the workload of the web server, Arduino translates the voltage to the data only when polled. We used Restful web communication to send the data from Arduino to the server. An Arduino program is called sketch, which has mainly two parts, initialization and the main operation.

Figure 8 shows a sample Arduino program segment. It first sets up the Ethernet connection and configures the timer using setTIme variable. The next part is the main operation in a loop form. Within the loop, the timer variable t keeps reading the data using getVoltage( ) every second. Figure 9 shows the code segment that transforms the voltage to temperature.
The `getVoltage()` function reads an analog signal from an analog input pin. The voltage is converted to temperature in Fahrenheit or Celsius in `getData()`.

The data, which consists of the server IP address, sensor ID, and the measured value, is sent to the server by `restConnection()`. The `restConnection()` sends data to a destination URL over REST (Representational State Transfer) (Roy, 2000). REST is created for networking applications, which consists of several constraints to address separation of concerns, visibility, reliability, scalability, flexibility, and performance as an architectural style (Hongjun, Li, 2011). In the next session, we show the RESTful web service for the sensor network.

6. RESTful Web Service

Traditional web service, Simple Object Access protocol (SOAP), is not suitable for sensor as due to the various model of the sensor application (Alghamdi, Lasebae, & Aiash, 2013). Due to the increase number of sensor, first generation web service, SOAP, is presenting multiple limitations when it handles various devices. Therefore, RESTful web service is receiving spotlight by abundant researchers due to its efficiency on handling the sensor. Second generation web service, RESTful web service, has several benefits to handle the sensor as it is based on Resource Oriented Architecture (ROA).

REST based on the ROA can handle various models of sensor. REST includes four attributes of the ROA as it is created using ROA. First attribute of REST is addressability. All resources are represented by Uniform Resource Identifiers (URI). This implies that all sensors can be identified through the unique URI. Each URI must be used to handle each object. Therefore, REST uses the URI as if URI was the variable of object oriented.

The `getVoltage()` function reads an analog signal from an analog input pin. The voltage is converted to temperature in Fahrenheit or Celsius in `getData()`. The `restConnection()` sends data to a destination URL over REST (Representational State Transfer) (Roy, 2000). REST is created for networking applications, which consists of several constraints to address separation of concerns, visibility, reliability, scalability, flexibility, and performance as an architectural style (Hongjun, Li, 2011). In the next session, we show the RESTful web service for the sensor network.

6.1. Basic Attributes of REST

**Figure 10. NevCAN with RESTful URI**

- http://NevCANIPAddress/Data
- http://NevCANIPAddress/TempIPAddress/Data
- http://NevCANIPAddress/TopoIP/Data
- http://NevCANIPAddress/TowerIP/Data
- http://NevCANIPAddress/TimeIP/Data
- http://NevCANIPAddress/WeatherIP/Data
- http://NevCANIPAddress/WeatherIP/Data
- http://NevCANIPAddress/WeatherIP/Data
- http://NevCANIPAddress/WeatherIP/Data
‘client state. So, clients have to express their current state via URI. Requested resources are only used once. Therefore, the request does not impact next request. As there is no session, statelessness is good for balancing server loads. When one of servers is highly loaded than does other servers, the request is passed onto other servers without creating additional work. Moreover, other beneficial attributes of statelessness are scalability (Xinyang, Jianjing, & Ying, 2009) visibility and reliability.

6.2. REST vs. SOAP Performance Test

Until recently, web services based on Simple Object Access Protocol (SOAP) was commonly used as a middleware for accessing equipment remotely. However, REST was chosen because of the required technology to build a Web Service is simpler than SOAP (Franco, et al., 2010). REST performs better in a sensor network compared to SOAP, and equipment vendors now recommends REST for applications consuming embedded resources. Figure 11 shows that Web Services based on REST is faster than SOAP when service or operation is requested, such as Register a client (REG), Send a message (SEND), Receive all messages (RCV), or Unregister a client (UREG) (Alhikisalo, 2012).

Typically more than one sensor data is collected by the sensors in every polling interval depending on the researchers needs. The sensor data is consequently passed to collectors, such as Arudino or CR1000. In order to properly process the accumulated data, the memory size of data logger should be sufficiently large. The message size also plays an important role. REST-based web service is again ideally positioned for remotely accessing collectors.

However, there are authentication problems due to statelessness of REST. REST does not utilize session between server and client. Therefore, client authentication process is necessary for REST whenever servers receive the clients’ requests. In the next session, we explain authentications currently implied by web service companies such as, HTTP Basic Auth, HTTP Digest Authentication, API Key, Access Token and OAuth, for RESTful web service.

7. ID-BASED AUTHENTICATION FOR REST

7.1. Current Authentication Schemes for Restful Web Service

RESTful web service constantly struggles with authentication process of users. While processing user authentication, statelessness can overload CPU and waste abundant amount of network resources. Servers have to authenticate the client every request. Instead of sending ID and password via post method, most of companies are trying to solve the problem by using HTTP Basic Auth, HTTP Digest Authentication, API Key, Access Token and OAuth. In this session, we show the authentication methods that are currently used.

7.1.1. HTTP Basic Authentication

It uses the ID and password of client to authenticate them in HTTP header. Figure 12 shows HTTP Authentication phases. When server requests authentication of client, server sends a message “HTTP 401 Not Authorized” with WWW-Authorized HTTP Header. Client’s ID and password gets encoded with Base64 and is stored in the HTTP Authentication header. It is not encrypted or hashed. So, it is usually used over HTTPS or SSL. However, HTTP Basic Authentication has critical issue that it doesn’t support logout function. Also, it has security vulnerability from replay attack, injection attack and middleware hijacking (Masood, 2013). Due to saving ID and password at HTTP Header, HTTP Basic Authentication can be exposed by the attack.

7.1.2. HTTP Digest Authentication

It is advanced HTTP Basic Authentication. HTTP Digest Authentication encrypt client’s ID and password via hash such as MD5. It can prevent hash functions from Rainbow Table attack by creating nonce from client. Also, it is secure from Replay Attack by creating timestamp from server. However, it has several vulnerable to security. It could be attacked by Man-in-the-middle Attack. Because, HTTP Digest Authentication does not provide confirmation server method to client. So, attacker can change the HTTP Digest Authentication to HTTP Basic Authentication. Also, due to HTTP Digest Authentication hash ID and password as well as...
nonce and realm, it is hard to use strong hash method such as bcrypt.

7.1.3. Token based Authentication (OAuth)

It is use a Token instead of ID and password. Figure 13 shows how to work the Token based Authentication for authentication client. Service Server (SS) redirect client’s browser when client request service to SS. Client login to Authorization Server (AS) to get Token. SS get Token from the AS. SS can authenticate using the Token. Client does not be exposed client’s ID and password to third party due to Token. So, Web Service companies such as Twitter, Yahoo, Google, Facebook, Microsoft and so on, using the method.

7.1.4 Current Authentication System Problem in Sensor Network

These three methods are mainly used for authentication from web service companies. However, HTTP Basic Authentication and HTTP Digest Authentication need session to keep the authentication. That means these two authentication methods are not statelessness due to session. Server has to store a session ID to keep session. Or, HTTP Basic Authentication and HTTP Digest Authentication have to authenticate each request from client, if these two methods do not use session. It makes wasted time and resource such as CPU, Network Resource. Also, it has security vulnerability due to contained ID and password. Most of company use OAuth instead of HTTP Basic Authentication and HTTP Digest Authentication due to authentication problem of these two methods. However, OAuth also has security vulnerability. It is Covert Redirect Vulnerability (CRV). It use redirection in figure 13 by explore parameter. Attacker forges login page (figure 13. ID and password) or redirection Uniform Resource Locator (URL) to get client’s ID and client’s password.

To solve these problems, we use ID-based Cryptography for authentication of RESTful web service. In the next session, we explain basic algorithm of ID-based Encryption (IBE). And then, we show the Boneh-Franklin scheme which is an IBE system proposed by Dan Boneh and Matthew K. franklin in 2001 (D.Boneh. & M. Franklin, 2001).

http://hipore.com/ijsc
7.2 Basic Concept of ID-based Encryption

IBE is an asymmetric key algorithm. It is first time implemented by Adi Shamir in 1984 (Adi, 1984). He developed IBE using email-address based Public Key Infrastructure (PKI). The email-address is unique information for the identity of user. Also, email-address is usually exposed to outside like public key. So, it can be used for public key generation source and identity. Figure 14 shows a work flow of IBE. When Bob and Alice want to communicate, they receive their private key which corresponding their public key from Private Key Generator (PKG). PKG is a trusted third party. PKG compute the clients (Alice and Bob) private key corresponding with their email. Also, using the PKG’s public key, any party can generate public key corresponding to the email by combining the PKG’s public key. So, Alice encrypt message using Bob’s public key generated via PKG’s public key. And she sends it to Bob. Bob can decrypt using own private key generated via PKG’s private key. The cryptographic processes are described in 3 steps below. Refer to the Table III for the definitions of the terms.

Step 1: PKG generate Alice and Bob private key using their email.

\[ K_{\text{Alice pri}} = G(MK_{\text{pri}}, \text{Alice email}) \]
\[ K_{\text{Bob pri}} = G(MK_{\text{pri}}, \text{Bob email}) \]

Step 2: Alice generate Bob’s public key via MK_{pub} with Bob_email. And, Alice encrypt message (m) using Bob’s public key. Alice send the cipher text to Bob

\[ K_{\text{Bob pub}} = G(MK_{\text{pub}}, \text{Bob email}) \]
\[ C = E(K_{\text{Bob pub}}, m) \]

Step 3: Bob decrypt cipher text to get message using Bob’s private key.

\[ m = D(K_{\text{Bob pri}}, C) \]

In the RESTful web service, we can use URI instead of email address. So, we can authentication via unique URI.

7.3 ID-based Authentication for REST

ID-based Authentication (IDA) is based on ID-based signatures with REST URI. We use the ID-based cryptography and REST unique URI for authentication. As a figure 10, all sensors can be represented by unique URI. Also, IBE use some unique information about the identity of the user such as email address, for public-key encryption. Therefore, we use a REST URI to authentication with IBE.

To make the IDA system, we make two tables to authenticate sensor. One is Key table in Asset Manager (AM). Key table store the sensor’s private key and URL. The Key table is used by making
sensors public key. So, the AM work like a PKG. And the URL is used as an unique information for the identity of user instead of email address.

The other table is Match table in Profile Manager (PM). The Match table checks user permissions for sensor. It limits the user access from sensor.

In the next section, we explain detail rule of AM and PM. Also, we show the extended architecture of the Web Service with AM, PM and sensor network. We enhanced basic Web Service architecture so that it can be more flexible and scalable Web Service for various equipment and users.

8. WEB SERVICE FOR SENSOR NETWORK

Web Service is frequently used for communicating to sensors and collectors. In this section, we will show our Extend Web Service architecture to handle sensor data. Typically, Web Service is consisted of Client, Middle, and Server (Botts & Robin, 2007). User interface is handled by Client tier. Users get data service via the web page. Middle tier works as a mediator. It handles the requests of the users. For example, the requested data by the users come from the Server tier, then gets processed by the Middle tier, and is sent to the users (Xuelei, Jia, & Bilan, 2009).

8.1 Extended Web Service Architecture with Sensor Network

In our project, we extended the middle tier to control remote sensors and collectors. Figure 15 illustrates the Extended Web Architecture with Sensor Network. In general environmental monitor system, collector stores the measured data collected by sensors. The measured data is temporarily stored in Collector prior to being moved to Middle tier. While it is possible to move the sensor data directly to the database in Server tier, we avoid it for security reasons. Remote sensors are generally exposed to risks, such as being stolen, damaged, or modified. So, we can protect newest data by moving the storage place. All the data is sent to database in the Middle tier for data integrity, data verification and data classification. To overcome collector’s limitations, we delegate some functions of collector such as configuration parameters and processing in, the Middle tier. By delegating functions to server, collector (Arduino) can focus on gathering sensor data with full resource and essential functions. Additionally, the sensors and collectors can be upgraded transparently to the rest of the system.

Also, we extend Middle tier for scalability and flexibility. Most of other Web Service for environmental data portal researches focus on constructing Web Service suitable for one specific type of collector and sensor (Xianfeng, Chaoqiang, Kagawa & Raghavan, 2010; Bock, Crowell, Prawirotidjo & Jamason, 2008). However, Web Service for environmental sensor network in the real world uses various types of collectors and user’s device. For example, NCCP collects over fifty different types by NevCAN. Therefore, it is crucial to extend the Middle tier in order to control various devices. Consequently, we made two more parts in the Middle tier (Sungchul, et al., 2014). One is PM and the other is AM. Users, who have different permissions, request services via various devices such as PC, cell phone, and smart devices. Such devices are handled by the PM in the Middle tier. The roles of PM include processing, handling and describing user’s request. The AM is created for collectors and sensor data. The sensor data from collectors such as CR1000, Arduino, and so on is processed by the AM. The sensor data is transformed into Sensor Model Language (SensorML) by the AM using URL. SensorML provides a framework to XML for standardizing data, capabilities, and systems of sensor (Botts & Robin, 2007). Also, the AM controls the collectors using command orders such as, time, unit, update, and cancel via the URL (Quint, 2003). So, users with permissions to control

Figure 16. Flow Web Service with Sensor Network

http://hipore.com/ijsc
the collector can control through client tier. The order command is received by FM and passed onto AM to handle collector.

8.2 Flow Web Service

We follow Model-View-Control (MVC) design pattern to build Web Service (Xuelei, Jia, & Bilan, 2009). Figure 16 shows the interactions among the modules (Sungchul, Juyeon, & Yoohwan, 2014). They work in the following sequence.

1. Controller programmed in Servlet receives client’s request by web page.
2. Current status of Model is constantly checked and updated by Controller’s commands.
3. Whenever the status is changed, view and Controller get the updated status of Model programmed in Java bean.
4. View produces the updated output. And controller changes the available set of commands by Model’s notification.
5. View sends the data from Model and associated formatting parameters to a 3rd party service (Google chart server).
6. 3rd party service (Google chart server) makes a graph image in Scalable Vector Graphics (SVG) format using the data and sends to View for visualization data.

In the next session, we show the data visualization. The data visualization represent chart to client. We use Google Charts to visualize the data.

9. DATA VISUALIZATION

We use Google chart to visualize the NCCP data. Figure 17 is an example of Google chart image in SVG format created with the actual NCCP sensor data. SVG is an XML-based vector image format for two-dimensional graphics, interactivity, and animation (Ying, 2012). So, the SVG image can be recreated depending on the data period or filter selection by users without additional connection any server. Google supports various types of chart such as line, pie, tree map and 3D. Therefore, developer doesn’t need an additional effort to make different types of data with same data. Also, it has dashboards for a high level interaction with users. The dashboard has a wide selection of rendering features for interacting user action such as selecting data period. Additionally, developers who use Google Chart don’t need to consider end-user’s devices. It is because of cross-browser compatibility, e.g., for IE, or Firefox, and cross-platform portability, e.g., for iPad, iPhone or Android (Ying, 2012).

Google Chart can directly use raw data and also utilize DataTable for making a chart. DataTable class sends the transformed data to visualization engine (Sakamoto, Matsumoto & Nakamura, 2012). The DataTable can be used for switching different types of charts and dashboards. Also, developer can easily implement sorting, modifying, finding (max, min) and filtering functions using DataTable. Additionally, Google Chart can use Google app such as Google Spreadsheets and Fusion Tables via DataTable (Sungchul, Juyeon, & Yoohwan, 2014).

10. CONCLUSION AND FUTURE WORKS

This research has designed an architectural model with Web Service to accommodate data collection and sharing needs in Nevada Nexus project. The performance of sensor devices has been reviewed and we learned the performance of Arduino is comparable to a traditional data logging device such as CR1000, thereby offering a more cost-effective

http://hipore.com/ijsc
solution. The Web architecture is designed based on MVC pattern. We used Restful Web Service to handle the sensor data sent by Arduino-based sensors and employed Google Charts service for visualizing data. With the web service architecture developed in this research, the data created from the environmental monitoring systems can be effectively shared among all researchers.

In the future, we plan to develop AM and PM in the Middle tier to interface with a large range of client devices and sensor devices. Furthermore, we need to investigate the security and confidentiality issues as REST-based Web Service may have some security concerns (Gabriel, et al., 2012) inherent to basic URL-based service.

11. ACKNOWLEDGEMENT

This material is based upon work supported by the National Science Foundation under grant number IIA-1301726.

12. DISCLAIMER

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

13. REFERENCES


Hiro, Gabriel., Cerqueira, Ferreira., Edna, Dias, Canedo., & Rafael, Timóteo, de, Sousa, Junior. (2013). IoT Architecture to Enable Intercommunication Through REST API and UPnP Using IP, ZigBee and Arduino. 1st International Workshop on Internet of Things Communications and Technologies (IoT’13), October, 7-9 2013, pp. 53 – 60


http://hipore.com/ijsc
Ju-Yeon Jo received the PhD degree in Computer science from Case Western Reserve University, Cleveland, Ohio. She is an associate professor of computer science department at the University of Nevada, Las Vegas where she joined in August 2006. From 2003 to 2006, she was an assistant professor of computer science department at California State University, Sacramento. Prior to that she was a member of technical staff at Lucent Technologies, Bell Labs, in Holmdel, New Jersey, and a software engineer at Coree Networks, a New Jersey based start-up company. Her current research interests include big data management and security, network security, networking protocol design and performance analysis, and secure software development.

Yooohwan Kim is an Associate Professor of Computer Science at University of Nevada, Las Vegas. He received his Ph.D. degrees in Computer Engineering from Case Western Reserve University in 2004 and his Bachelor degree in Economics from Seoul National University, Korea in 1989. Between 1997 and 1999 he was a Member of Technical Staff at Lucent Technologies, Whippany, NJ, developing wireless telecommunication software. His research interests include secure network protocol design, network authentication, SCADA and Smart Grid security.

Sungchul Lee was born in Seoul, Korea. He received his M.Sc degree in computer science from the University of Nevada, Las Vegas (UNLV) in 2012. He has been in the Ph.D. program in computer science at UNLV since 2013. He is one of the members of the Nevada Energy-Water-Environment Nexus Project funded by NSF where he is conducting the data analysis and data security research. His research interests include big data analytics with Hadoop system, secure network protocol design, authentication and access control schemes, and user behavior analysis in Smart Grid environment. He is also investigating navigation and collision avoidance algorithms for Unmanned Aerial Systems (UAS) through simulations studies.


Authors

Sungchul Lee

Ju-Yeon Jo

Yooohwan Kim

http://hipore.com/ijsc