Weather Condition Double Checking in Internet SCADA Environment

Tai-hoon Kim
Multimedia Engineering Department,
Hannam University
133 Ojeong-dong, Daeduk-gu, Daejeon,
Korea
taihoonn@hnu.kr

Abstract: - The function of SCADA is collecting of the information, transferring it back to the central site, carrying out any necessary analysis and control and then displaying that information on a number of operator screens. Systems automatically control the actions and control the process of automation. Internet SCADA was developed to widen the coverage span of the SCADA system. In this paper, we design a double checking scheme for Weather Condition in Internet SCADA Environment. This is to improve the accuracy of data and to improve the performance of SCADA systems.

Key-Words: - SCADA, TeleControl, API, Control Systems

1 Introduction

Supervisory Control and Data Acquisition (SCADA) systems are now used in mining industries, modern manufacturing and industrial processes, public and private utilities, leisure and security industries. In these situations, telemetry is needed to connect systems and equipment separated by long distances. Some of this ranges to up to thousands of kilometers. In these cases, SCADA are sometimes connected through the Internet. The purpose of this is to widen the coverage span of the SCADA System. To improve the accuracy of data and to improve the performance of SCADA systems, we design a double checking scheme for Weather Condition in Internet SCADA Environment. This scheme uses data from weather API Providers. Many API Provider such as Google, Yahoo, etc have Weather API's. Weather API's can give weather condition and forecast about a specific place.

2 Related Technologies

In this section, we discuss the related technologies for this work. SCADA or Supervisory Control and Data Acquisition Systems and its components are discussed. The integration of SCADA to the Internet is studied and lastly, the API's and its functionality.

2.1 Supervisory Control and Data Acquisition Systems

Supervisory Control and Data Acquisition (SCADA) existed long time ago when control systems were introduced. SCADA systems that time use data acquisition by using strip chart recorders, panels of meters, and lights. Not similar to modern SCADA systems, there is an operator which manually operates various control knobs exercised supervisory control. These devices are still used to do supervisory control and data acquisition on power generating facilities, plants and factories.

Telemetry is automatic transmission and measurement of data from remote sources by wire or radio or other means. It is also used to send commands, programs and receives monitoring information from these remote locations. SCADA is the combination of telemetry and data acquisition. Supervisory Control and Data Acquisition system is compose of collecting of the information, transferring it to the central site, carrying out any necessary analysis and control and then displaying that information on the operator screens. The required control actions are then passed back to the process. [1]. Typically SCADA systems include the following components: [2]

1. Operating equipment such as pumps, valves, conveyors and substation breakers that can be controlled by energizing actuators or relays.
2. Local processors that communicate with the site's instruments and operating equipment.
3. Instruments in the field or in a facility that sense conditions such as pH, temperature, pressure, power level and flow rate.
4. Short range communications between the local processors and the instruments and operating equipment.
5. Long range communications between the local processors and host computers.
6. Host computers that act as the central point of monitoring and control.

The measurement and control system of SCADA has one master terminal unit (MTU) which could be called the brain of the system and one or more remote terminal units (RTU). The RTUs gather the data locally and send them to the MTU which then issues suitable commands to be executed on site. A system of either standard or customized software is used to collate, interpret and manage the data.

Supervisory Control and Data Acquisition (SCADA) is conventionally set upped in a private network not connected to the internet. This is done for the purpose of isolating the confidential information as well as the control to the system itself. Because of the distance, processing of reports and the emerging technologies, SCADA can now be connected to the internet. This can bring a lot of advantages and disadvantages which will be discussed in the sections.

Conventionally, relay logic was used to control production and plant systems. With the discovery of the CPU and other electronic devices, manufacturers incorporated digital electronics into relay logic equipment. Programmable logic controllers or PLC's are still the most widely used control systems in industry. As need to monitor and control more devices in the plant grew, the PLCs were distributed and the systems became more intelligent and smaller in size. PLCs (Programmable logic controllers) and DCS (distributed control systems) are used as shown in Figure 1.

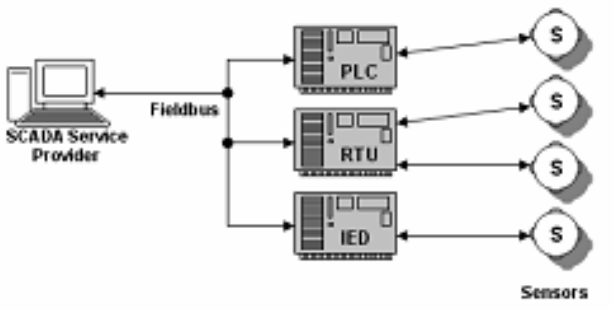

Figure 1. Conventional SCADA Architecture

Data acquisition begins at the RTU or PLC level and includes meter readings and equipment status reports that are communicated to SCADA as required. Data is then compiled and formatted in such a way that a control room operator using the HMI can make supervisory decisions to adjust or override normal RTU (PLC) controls. Data may also be fed to a Historian, often built on a commodity Database Management System, to allow trending and other analytical auditing.

SCADA systems typically implement a distributed database, commonly referred to as a tag database, which contains data elements called tags or points. A point represents a single input or output value monitored or controlled by the system. Points can be either "hard" or "soft". A hard point represents an actual input or output within the system, while a soft point results from logic and math operations applied to other points. (Most implementations conceptually remove the distinction by making every property a "soft" point expression, which may, in the simplest case, equal a single hard point.) Points are normally stored as value-timestamp pairs: a value, and the timestamp when it was recorded or calculated. A series of value-timestamp pairs gives the history of that point. It's also common to store additional metadata with tags, such as the path to a field device or PLC register, design time comments, and alarm information.

2.1.1 SCADA Human Machine Interface

In SCADA and in the industrial design field of human-machine interaction, the user interface is (a place) where interaction between humans and machines occurs. The goal of interaction between a human and a machine at the user interface is effective operation and control of the machine, and

feedback from the machine which aids the operator in making operational decisions. Examples of this broad concept of user interfaces include the interactive aspects of computer operating systems, hand tools, heavy machinery operator controls and process controls.

The goal of human-machine interaction engineering is to produce a user interface which makes it easy, efficient, and enjoyable to operate a machine in the way which produces the desired result. This generally means that the operator needs to provide minimal input to achieve the desired output, and also that the machine minimizes undesired outputs to the human.

Ever since the increased use of personal computers and the relative decline in societal awareness of heavy machinery, the term user interface has taken on overtones of the (graphical) user interface, while industrial control panel and machinery control design discussions more commonly refer to human-machine interfaces.

The design of a user interface affects the amount of effort the user must expend to provide input for the system and to interpret the output of the system, and how much effort it takes to learn how to do this. Usability is the degree to which the design of a particular user interface takes into account the human psychology and physiology of the users, and makes the process of using the system effective, efficient and satisfying.

Usability is mainly a characteristic of the user interface, but is also associated with the functionalities of the product and the process to design it. It describes how well a product can be used for its intended purpose by its target users with efficiency, effectiveness, and satisfaction.

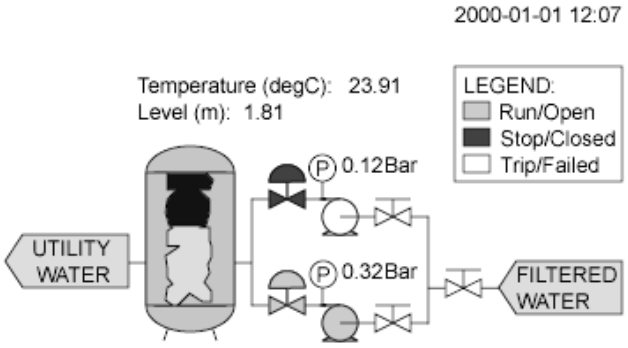

Figure 2. An Example of a SCADA Human Machine Interface

SCADA system includes a user interface which is usually called Human Machine Interface (HMI). The HMI of a SCADA system is where data is processed and presented to be viewed and monitored by a human operator. This interface usually includes controls where the individual can interface with the SCADA system. HMI's are an easy way to standardize the facilitation of monitoring multiple RTU's or PLC's (programmable logic controllers). Usually RTU's or PLC's will run a pre programmed process, but monitoring each of them individually can be difficult, usually because they are spread out over the system. Because RTU's and PLC's historically had no standardized method to display or present data to an operator, the SCADA system communicates with PLC's throughout the system network and processes information that is easily disseminated by the HMI. HMI's can also be linked to a database, which can use data gathered from PLC's or RTU's to provide graphs on trends, logistic info, schematics for a specific sensor or machine or even make troubleshooting guides accessible. In the last decade, practically all SCADA systems include an integrated HMI and PLC device making it extremely easy to run and monitor a SCADA system.

The HMI package for the SCADA system typically includes a drawing program that the operators or system maintenance personnel use to change the way these points are represented in the interface. These representations can be as simple as an on-screen traffic light, which represents the state of an actual traffic light in the field, or as complex as a multi-projector display representing the position of all of the elevators in a skyscraper or all of the trains on a railway.

Alarm handling is an important part of most SCADA implementations. The system monitors whether certain alarm conditions are satisfied, to determine when an alarm event has occurred. Once an alarm event has been detected, one or more actions are taken (such as the activation of one or more alarm indicators, and perhaps the generation of email or text messages so that management or remote SCADA operators are informed). In many cases, a SCADA operator may have to acknowledge the alarm event; this may deactivate some alarm indicators, whereas other indicators remain active until the alarm conditions are cleared. Alarm

conditions can be explicit - for example, an alarm point is a digital status point that has either the value NORMAL or ALARM that is calculated by a formula based on the values in other analogue and digital points - or implicit: the SCADA system might automatically monitor whether the value in an analogue point lies outside high and low limit values associated with that point. Examples of alarm indicators include a siren, a pop-up box on a screen, or a colored or flashing area on a screen (that might act in a similar way to the "fuel tank empty" light in a car); in each case, the role of the alarm indicator is to draw the operator's attention to the part of the system 'in alarm' so that appropriate action can be taken. In designing SCADA systems, care is needed in coping with a cascade of alarm events occurring in a short time, otherwise the underlying cause (which might not be the earliest event detected) may get lost in the noise. Unfortunately, when used as a noun, the word 'alarm' is used rather loosely in the industry; thus, depending on context it might mean an alarm point, an alarm event or an alarm indicator.

2.1.2 SCADA Hardware

A SCADA system consists of a number of remote terminal units (RTUs) collecting field data and sending that data back to a master station, via a communication system. The master station displays the acquired data and allows the operator to perform remote control tasks.

The accurate and timely data allows for optimization of the plant operation and process. Other benefits include more efficient, reliable and most importantly, safer operations. This results in a lower cost of operation compared to earlier non-automated systems.

Supervisory Control and Data Acquisition Systems usually have Distributed Control System components. PLCs or RTUs are also commonly used; they are capable of autonomously executing simple logic processes without a master computer controlling it. A functional block programming language, IEC 61131-3, is frequently used to create programs which run on these PLCs and RTUs. This allows SCADA system engineers to perform both the design and implementation of a program to be executed on an RTU or PLC. From 1998, major PLC manufacturers have offered integrated HMI/SCADA systems, many use open and non-proprietary communications protocols. Many third-party HMI/SCADA packages, offering built-in compatibility with most major PLCs, have also entered the market, allowing mechanical engineers, electrical engineers and technicians to configure HMIs themselves. Many other hardware are also basing its functionality to those of PLC's. [11]

The communications system provides the pathway for communication between the master station and the remote sites. This communication system can be wire, fiber optic, radio, telephone line, microwave and possibly even satellite. Specific protocols and error detection philosophies are used for efficient and optimum transfer of data.

The master station (or sub-masters) gather data from the various RTUs and generally provide an operator interface for display of information and control of the remote sites. In large telemetry systems, sub-master sites gather information from remote sites and act as a relay back to the control master station.

2.1.3 SCADA Software

Supervisory Control and Data Acquisition software can be divided into proprietary type or open type. Proprietary software are developed and designed for the specific hardware and are usually sold together. The main problem with these systems is the overwhelming reliance on the supplier of the system. Open software systems are designed to communicate and control different types of hardware. It is popular because of the interoperability they bring to the system. [1] WonderWare and Citect are just two of the open software packages available in the market for SCADA systems. Some packages are now including asset management integrated within the SCADA system.

2.1.4 SCADA Communication

SCADA systems have traditionally used combinations of radio and direct serial or modem connections to meet communication requirements, although Ethernet and IP over SONET / SDH is also frequently used at large sites such as railways and power stations. The remote management or

monitoring function of a SCADA system is often referred to as telemetry.

This has also come under threat with some customers wanting SCADA data to travel over their pre-established corporate networks or to share the network with other applications. The legacy of the early low-bandwidth protocols remains, though. SCADA protocols are designed to be very compact and many are designed to send information to the master station only when the master station polls the RTU. Typical legacy SCADA protocols include Modbus RTU, RP-570, Profibus and Conitel. These communication protocols are all SCADA-vendor specific but are widely adopted and used. Standard protocols are IEC 60870-5-101 or 104, IEC 61850 and DNP3. These communication protocols are standardized and recognized by all major SCADA vendors. Many of these protocols now contain extensions to operate over TCP/IP. It is good security engineering practice to avoid connecting SCADA systems to the Internet so the attack surface is reduced.

RTUs and other automatic controller devices were being developed before the advent of industry wide standards for interoperability. The result is that developers and their management created a multitude of control protocols. Among the larger vendors, there was also the incentive to create their own protocol to "lock in" their customer base. A list of automation protocols is being compiled here.

Recently, OLE for Process Control (OPC) has become a widely accepted solution for intercommunicating different hardware and software, allowing communication even between devices originally not intended to be part of an industrial network.

Central computer of the data acquisition system, located in the hydro power plant, provides measurements performance according to a preset program, the instrumentation existing at this time and remote communications by RS485 bus, using Master-Slave architecture and IEC1107, Modbus RTU, ASCII protocols.[10]

2.2 Internet SCADA

Conventional SCADA only have 4 components: the master station, plc/rtu, fieldbus and sensors. Internet SCADA replaces or extends the fieldbus to the internet. This means that the Master Station can be on a different network or location.

In Figure 3, you can see the architecture of SCADA which is connected through the internet. Like a normal SCADA, it has RTUs/PLCs/IEDs, The SCADA Service Provider or the Master Station. This also includes the user-access to SCADA website. This is for the smaller SCADA operators that can avail the services provided by the SCADA service provider. It can either be a company that uses SCADA exclusively. Another component of the internet SCADA is the Customer Application which allows report generation or billing. Along with the fieldbus, the internet is an extension. This is setup like a private network so that only the master station can have access to the remote assets. The master also has an extension that acts as a web server so that the SCADA users and customers can access the data through the SCADA provider website.

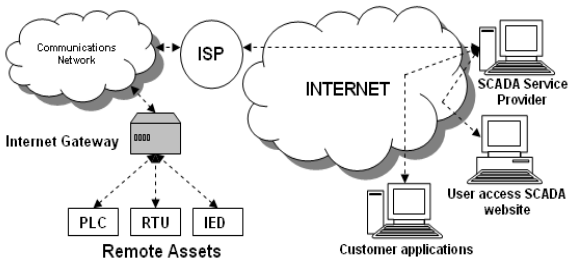

Figure 3. Internet SCADA Architecture [3]

One may ask why we need to connect SCADA on the Internet even though there are a lot of issues surrounding it. The answer is because of many advantages it presents. [4]

- Parallel Polling
- Redundancy and Hot Standby
- Large addressing range
- Wide area connectivity and pervasive
- Standardization
- Routable
- Integration of IT to Automation and Monitoring Networks

2.3 Web API

API's or application program interface, are set of

routines, protocols, and tools for building software applications. A good API makes it easier to develop a program by providing all the building blocks. A programmer then puts the blocks together. [5]

When used in the context of web development, an API is typically a defined set of Hypertext Transfer Protocol (HTTP) request messages, along with a definition of the structure of response messages, which is usually in an Extensible Markup Language (XML) or JavaScript Object Notation (JSON) format. While "Web API" is virtually a synonym for web service, the recent trend (so-called Web 2.0) has been moving away from Simple Object Access Protocol (SOAP) based services towards more direct Representational State Transfer (REST) style communications.[6] Web APIs allow the combination of multiple services into new applications known as mashups.[7]

The practice of publishing APIs has allowed web communities to create an open architecture for sharing content and data between communities and applications. In this way, content that is created in one place can be dynamically posted and updated in multiple locations on the web.[5]

2.3.4 Web Services

Web services are typically application programming interfaces (API) or web APIs that are accessed via Hypertext Transfer Protocol and executed on a remote system hosting the requested services. Web services tend to fall into one of two camps: Big Web Services and RESTful Web Services.

Big Web Services use Extensible Markup Language (XML) messages that follow the Simple Object Access Protocol (SOAP) standard and have been popular with traditional enterprise. In such systems, there is often a machine-readable description of the operations offered by the service written in the Web Services Description Language (WSDL). The latter is not a requirement of a SOAP endpoint, but it is a prerequisite for automated client-side code generation in many Java and .NET SOAP frameworks (frameworks such as Spring, Apache Axis2 and Apache CXF being notable exceptions). Some industry organizations, such as the WS-I, mandate both SOAP and WSDL in their definition of a web service.

Web API is a development in web services (in a movement called Web 2.0) where emphasis has been moving away from SOAP based services towards Representational State Transfer (REST) based communications. REST services do not require XML, SOAP, or WSDL service-API definitions.

Web APIs allow the combination of multiple web services into new applications known as mashups.

When used in the context of web development, web API is typically a defined set of Hypertext Transfer Protocol (HTTP) request messages along with a definition of the structure of response messages, usually expressed in an Extensible Markup Language (XML) or JavaScript Object Notation (JSON) format.

When running composite web services, each sub service can be considered autonomous. The user has no control over these services. Also the web services themselves are not reliable; the service provider may remove, change or update their services without giving notice to users. The reliability and fault tolerance is not well supported; faults may happen during the execution. Exception handling in the context of web services is still an open research issue.

The W3C defines a "web service" as "a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically Web Services Description Language WSDL). Other systems interact with the web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other web-related standards."

The W3C also states, "We can identify two major classes of web services, REST-compliant Web services, in which the primary purpose of the service is to manipulate XML representations of Web resources using a uniform set of "stateless" operations; and arbitrary Web services, in which the service may expose an arbitrary set of operations."

Most operating environments, such as MS-Windows, provide an API so that programmers can write applications consistent with the operating environment. Although APIs are designed for programmers, they are ultimately good for users because they guarantee that all programs using a common API will have similar interfaces. This makes it easier for users to learn new programs. [5]

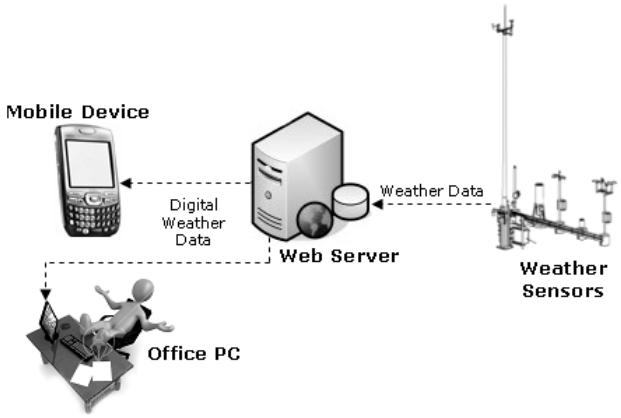

Figure 4. Data transmission of Weather API's

Many API Provider such as Google, Yahoo, etc have Weather API's. Weather API's can give weather condition and forecast about a specific place.

2.3.4 Example of API Implementations

-A Java API for Using a Native PGP [8]

The Java PGP API has been implemented within the Java class PGPi. This class provides the interface between Java applications and the C library. PGPi offers five different types of methods:

- Constructors
- Native methods
- Control methods
- Status methods
- Error handling

This API offers Java application developers access to a native implementation of the widely used public key encryption application PGP. This API makes it possible to use an approved encryption implementation not only for files and emails but also for network traffic. For most uses this API provides faster performance than the script-based solution because the encrypted and signed messages (software agents) sent over the network will be mostly smaller than 50kB. Furthermore, this API compiles and works under Windows, too. So there is no need to adapt the shell scripts and the output-parsing algorithm if one intends to use it on a different platform. The script-based solution needs to parse the output produced by PGP in order to determine whether a signature is valid, who is the signer and whether a message has been encrypted or just signed. This output can differ from platform to platform and it surely differs between different PGP versions.

-A WSDL-based service and its container[9]

The services deployed by the grid is described by a Web Service Description Language (WSDL) (http://www.w3c.org) document that allows a remote interface to be exposed. The document describes the service as a collection of communications endpoints or ports; it includes abstract and technical information. All the interactions are supported by messages transported within the SOAP protocol and the data being exchanged are specified as part of the message included in the SOAP document.

Every type of action allowed at an endpoint is considered an operation. Collections of operations possible on an endpoint are grouped together into port types. Messages, operations and port-types are all abstract definitions. Furthermore a port is defined by being associated with a network address with a reusable binding that is protocol and data format specified for a particular port type. The collections of ports define a service. Web Services are usually deployed in a web application container that is transformed from a container for presentation and tightly coupled logic to an infrastructure that equally supports asynchronous messages and flow coordination.

3. Integration of Technologies to Double Check Weather Condition

Weather API's can be integration to Internet SCADA systems to double check the weather condition. Weather sensors of SCADA systems may not gather correct data. This is very crucial and integration of API's can improve the data gathered.

Figure 5. SCADA Service Provider getting information from API service server

The following is the code that we use in gathering the weather information from Google Web API.

```
Dim xml_document As New DOMDocument
Dim xml_node As IXMLDOMNode
Dim pic_icon As String

Private Sub Timer1_Timer()

debugXML.Text = inet.OpenURL("API
source", icString)
xml_document.loadXML debugXML.Text
Set xml_node =
xml_document.documentElement.selectSin
gleNode("//xml_api_reply/weather/forec
ast_information/city")
Form1.Caption = "Weather Report: " +
xml_node.Attributes.getNamedItem("data
").Text
Set xml_node =
xml_document.documentElement.selectSin
gleNode("//xml_api_reply/weather/forec
ast_information/forecast_date")
txtDate.Caption = Now
Set xml_node =
xml_document.documentElement.selectSin
gleNode("//xml_api_reply/weather/curre
nt_conditions/temp_c")
txtTempC.Caption =
xml_node.Attributes.getNamedItem("data
").Text

If Val(txtTempC.Caption) > 28 Then
Shape1.FillColor = &HFF&
Shape2.FillColor = &HFF&
Label2.Caption = "Please dont forget to
bring umbrella. Its hot outside"

ElseIf Val(txtTempC.Caption) < 15 Then

Shape1.FillColor = &HFF0000
Shape2.FillColor = &HFF0000
Label2.Caption = "Its cold outside.
Bring your coat"

Else

Shape1.FillColor = &HC0FFFF
Shape2.FillColor = &HC0FFFF
Label2.Caption = "We have a fair weather.
Just bring umbrella"

End If

Set xml_node =
xml_document.documentElement.selectSin
gleNode("//xml_api_reply/weather/curre
nt_conditions/temp_f")
txtTempF.Caption =
xml_node.Attributes.getNamedItem("data
").Text
Set xml_node =
xml_document.documentElement.selectSin
gleNode("//xml_api_reply/weather/curre
nt_conditions/humidity")
txtHumid.Caption =
xml_node.Attributes.getNamedItem("data
").Text
Set xml_node =
xml_document.documentElement.selectSin
gleNode("//xml_api_reply/weather/curre
nt_conditions/wind_condition")
txtWind.Caption =
xml_node.Attributes.getNamedItem("data
").Text
Set xml_node =
xml_document.documentElement.selectSin
gleNode("//xml_api_reply/weather/curre
nt_conditions/icon")
pic_icon = App.Path +
xml_node.Attributes.getNamedItem("data
").Text
picIcon.Picture = LoadPicture(pic_icon)

End Sub

Private Sub Label1_Click()

If Label1.Caption = "Show tommorow's
forecast" Then

Dim low As String
```

```
Dim high As String
Dim flow As Integer
Dim fhigh As Integer

txtDate.Caption = DateAdd("d", 1, Now)

Set xml_node =
xml_document.documentElement.selectSin
gleNode("//xml_api_reply/weather/forec
ast_conditions/low")
low =
xml_node.Attributes.getNamedItem("data
").Text
Set xml_node =
xml_document.documentElement.selectSin
gleNode("//xml_api_reply/weather/forec
ast_conditions/high")
high =
xml_node.Attributes.getNamedItem("data
").Text

flow = (5 / 9) * (Val(low) - 32)
fhigh = (5 / 9) * (Val(high) - 32)

txtTempF.Caption = low + "~" + high
txtTempC.Caption = flow & "~" & fhigh


txtWind.Caption = ""

Set xml_node =
xml_document.documentElement.selectSin
gleNode("//xml_api_reply/weather/forec
ast_conditions/condition")
txtHumid.Caption = "Condition: " &
xml_node.Attributes.getNamedItem("data
").Text
Set xml_node =
xml_document.documentElement.selectSin
gleNode("//xml_api_reply/weather/forec
ast_conditions/icon")
pic_icon = App.Path +
xml_node.Attributes.getNamedItem("data
").Text

If Dir(pic_icon) <> "" Then
picIcon.Picture = LoadPicture(pic_icon)
Else
picIcon.Picture = LoadPicture("")
End If

Timer1.Enabled = False

Label1.Caption = "Back to current
conditions"

Else

Label1.Caption = "Show tommorow's
forecast"
Timer1.Enabled = True
End If
End Sub
```

Here is the source API.

```
<xml_api_reply version="1">

<weather   module_id="0"   tab_id="0"
mobile_row="0"       mobile_zipped="1"
row="0" section="0">

<forecast_information>
<city data="daejeon,kr"/>
<postal_code data="daejeon,kr"/>
<latitude_e6 data=""/>
<longitude_e6 data=""/>
<forecast_date data="2009-06-09"/>
<current_date_time    data="2009-06-09
02:20:00 +0000"/>
<unit_system data="US"/>
</forecast_information>

<current_conditions>
<condition data="Overcast"/>
<temp_f data="70"/>
<temp_c data="21"/>
<humidity data="Humidity: 78%"/>
<icon
data="/ig/images/weather/cloudy.gif"/>
<wind_condition  data="Wind:  E  at  4
mph"/>
</current_conditions>

<forecast_conditions>
<day_of_week data="Tue"/>
<low data="64"/>
<high data="75"/>
<icon
data="/ig/images/weather/chance_of_rai
n.gif"/>
<condition data="Chance of Rain"/>
```

```
</forecast_conditions>

</weather>

</xml_api_reply>
```

The result application of the above code can be found in the Figure 6. This application has the function of gathering weather information such as temperature, Humidity, Wind direction, etc.

Figure 6. Interface of an application that we created to gather weather information of a specific place from an API source.

SCADA controller or SCADA master station can get both data from the sensor (x) and the data from the Weather API (y). Usually, the controller only bases the commands on the sensor data. Since we integrate the Weather API to the system, we can also gather its data and we propose to get the average between the Sensor data and the API's data to get the base data (z) in which the commands will be based.

$$z = (x + y) / 2 \qquad (1)$$

Formula (1) will be the bases of the SCADA Controller in executing commands to the remote terminals. In Figure 7, we can see the comparison between the gathered Sensor data, API data and the average data. We will notice that there's sometimes a difference between the Sensor data and API data.

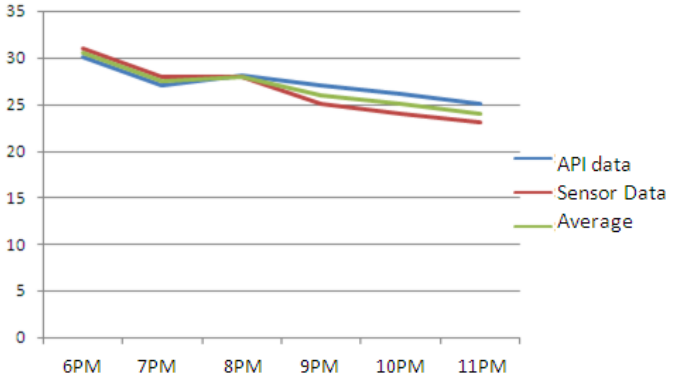

Figure 7. Comparisons of Gathered Sensor Data, API Data and the Average

4. Future Studies

Existing and future security issues may occur while implementing this architechture. In our future studies we plan to eliminate security issues is SCADA and in Web Services such as the following:

4.2 SCADA Security Issues

Even before SCADA was connected to the Internet, It is already surrounded by many security Issues and now the Internet has made them more vulnerable to attacks. Consequently, the security of SCADA-based systems has come into question as they are increasingly seen as extremely vulnerable to cyberwarfare/cyberterrorism attacks. Here are the common security issues in SCADA: [12]

- The lack of concern about security and authentication in the design, deployment and operation of existing SCADA networks.
- The belief that SCADA systems have the benefit of security through obscurity through the use of specialized protocols and proprietary interfaces.
- The belief that SCADA networks are secure because they are purportedly physically secured.
- The belief that SCADA networks are secure because they are supposedly disconnected from the Internet.
- IP Performance Overhead of SCADA connected to the Internet.

SCADA systems are used to control and monitor physical processes, examples of which are

transmission of electricity, transportation of gas and oil in pipelines, water distribution, traffic lights, and other systems used as the basis of modern society. The security of these SCADA systems is important because compromise or destruction of these systems would impact multiple areas of society far removed from the original compromise. For example, a blackout caused by a compromised electrical SCADA system would cause financial losses to all the customers that received electricity from that source. How security will affect legacy SCADA and new deployments remains to be seen.

There are two distinct threats to a modern SCADA system. First is the threat of unauthorized access to the control software, whether it be human access or changes induced intentionally or accidentally by virus infections and other software threats residing on the control host machine. Second is the threat of packet access to the network segments hosting SCADA devices. In many cases, there is rudimentary or no security on the actual packet control protocol, so anyone who can send packets to the SCADA device can control it. In many cases SCADA users assume that a VPN is sufficient protection and are unaware that physical access to SCADA-related network jacks and switches provides the ability to totally bypass all security on the control software and fully control those SCADA networks. These kinds of physical access attacks bypass firewall and VPN security and are best addressed by endpoint-to-endpoint authentication and authorization such as are commonly provided in the non-SCADA world by in-device SSL or other cryptographic techniques.

Many vendors of SCADA and control products have begun to address these risks in a basic sense by developing lines of specialized industrial firewall and VPN solutions for TCP/IP-based SCADA networks. Additionally, application whitelisting solutions are being implemented because of their ability to prevent malware and unauthorized application changes without the performance impacts of traditional antivirus scans.

4.2 Web Services Security Issues and Criticisms

Critics of non-RESTful Web services often complain that they are too complex and based upon large software vendors or integrators, rather than typical open source implementations. There are open source implementations like Apache Axis and Apache CXF.

One key concern of the REST Web Service developers is that the SOAP WS toolkits make it easy to define new interfaces for remote interaction, often relying on introspection to extract the WSDL, since a minor change on the server (even an upgrade of the SOAP stack) can result in different WSDL and a different service interface[8]. The client-side classes that can be generated from WSDL and XSD descriptions of the service are often similarly tied to a particular version of the SOAP endpoint and can break if the endpoint changes or the client-side SOAP stack is upgraded. Well-designed SOAP endpoints (with handwritten XSD and WSDL) do not suffer from this but there is still the problem that a custom interface for every service requires a custom client for every service.

There are also concerns about performance due to Web services' use of XML as a message format and SOAP/HTTP in enveloping and transport. However, emerging XML parsing and indexing technologies, such as VTD-XML, promise to address XML-related performance issues.

5. Conclusion

SCADA systems are very important since they control most infrastructures that we consider critical. The security of these SCADA systems is important because compromise or destruction of these systems would impact multiple areas of society far removed from the original compromise. The data that is gathered by the system is very important. The system reacts to the data it gets. Imagine what will happen if the data is not accurate. It can damage the society. To improve the accuracy of data and to improve the performance of SCADA systems, we design a double checking scheme for Weather Condition in Internet SCADA Environment. This scheme uses data from weather API Providers. Many API Provider such as Google, Yahoo, etc have Weather API's. Weather API's can give weather condition and forecast about a specific place.

References

1. D. Bailey and E. Wright (2003) Practical SCADA for Industry
2. Andrew Hildick-Smith (2005) Security for Critical Infrastructure SCADA Systems
3. D. Wallace (2003) Control Engineering. How to put SCADA on the Internet http://www.controleng.com/article/CA321065.html Accessed: January 2009
4. Internet and Web-based SCADA http://www.scadalink.com/technotesIP.htm Accessed: January 2009
5. "What is API? A word definition from the Webopedia Computer Dictionary" http://www.webopedia.com/TERM/A/API.html Accessed: April 2010
6. Benslimane, Djamal; Schahram Dustdar, and Amit Sheth (2008). "Services Mashups: The New Generation of Web Applications". IEEE Internet Computing, vol. 12, no. 5. Institute of Electrical and Electronics Engineers. pp. 13–15.
7. Niccolai, James (2008-04-23), "So What Is an Enterprise Mashup, Anyway?", PC World
8. THOMAS JAMPEN, MANUEL GU¨ NTER, TORSTEN BRAUN, "A Java API for Using a Native PGP Implementation", 6th WSEAS CSCC (CSCC 2002) MULTICONFERENCE
9. SERENA PASTORE, "Internet technologies and the grid paradigm: designing a custom environment for web service-based applications", Proceedings of the 6th WSEAS International Conference on Simulation, Modelling and Optimization, Lisbon, Portugal, September 22-24, 2006, pp. 693-698
10. COSTIN CEPISCA, HORIA ANDREI, EMIL PETRESCU, CRISTIAN PIRVU, CAMELIA PETRESCU, "Remote Data Acquisition System for Hydro Power Plants", Proceedings of the 6th WSEAS International Conference on Power Systems, Lisbon, Portugal, September 22-24, 2006, pp. 59-64
11. RAMÓN MARTÍNEZ-RODRÍGUEZ-OSORIO, MIGUEL CALVO-RAMÓN, MIGUEL Á. FERNÁNDEZ-OTERO, LUIS CUELLAR NAVARRETE, "Smart control system for LEDs traffic-lights based on PLC", Proceedings of the 6th WSEAS International Conference on Power Systems, Lisbon, Portugal, September 22-24, 2006, pp. 256-260
12. Rosslin John Robles, MIn-kyu Choi, Eun-suk Cho, "A Paradigm Solution to P2P Security Issues," AST 2009,pp.3-7, The First International e-Conference on Advanced Science and Technology, 7-9 March 2009, ISBN 978-0-7695-3672-9