Server Watch.

BTech 450DT – End-of-Semester Project Report

Anish Doshi

ados001@ec.auckland.ac.nz

Supervisor: S. Manoharan

Industry Mentor: Sean Tindale

4th June, 2007.

Department Of Computer Science.

University Of Auckland.

Contents

Abstract

The value of a service provided by a server is very high. This makes it very crucial to monitor the servers when they are running a valuable service. When such a server breaks down, the cost of not providing the service is very high. There are many different ways in which this problem can be tackled. But these servers need to get fully functional and start providing it's as soon as possible.

For this project, we create a system which will monitor the health of a server at regular intervals. This information can be viewed from a web browser as well as a mobile application. Monitoring a server through a mobile phone gives the administrator the power to monitor the server from any part of the world where he can access a mobile network. Also, this system adds the functionality to alert the user by sending warning messages through SMS (Short Messaging Service).

1 Introduction

When computer systems grew in size, the concept of distributed systems was invented. Distributed computing is a method of running different parts of the application on different computers, which connect each other over a network (Distributed computing, n.d.). This gave birth to the idiom of client/server architecture. Server is a computer or software that runs on a computer that provides specific kind of service to client software running on the same computer or other computers on the network. Client is a computer or software on a computer that requests utilize the service served by a server. Also, the server application can be distributed over different computers.

The services served by a server are the most valuable resources for some systems and eventually for the organization. E.g. A web hosting server, a banking system or a mail server or a database server. The organizations depending on these services suffer a setback when these servers go down. There are many ways to monitor this. But they are all traditional ways in which the servers are monitored using an application that resides on the server or on the network that the server is on. The server administrator needs to be on a computer to check the health of the system. The other way is when some one whose client depends on the server calls the administrator and notifies him. This is of course not a preferred method.

Here, we first do a requirement analysis of the system. We will understand different elements of a server that we will monitor through this system. Then we will look at a few technologies that can be implemented. Then we will look at the design architecture of some parts of the system.

1.1 The Project

The aim of this project is to design and deliver a system which can monitor a windows based server using a browser as well as a mobile device. Also, this system is should be able to deliver alert messages to the user in the case of an emergency such as server shutdown or high usage of resources for an abnormally long period.

This system has to designed and implemented from scratch. In the start, a research of the possibilities and scope of such a monitoring system has to be done. Then the technologies that be used for this system have to be researched. The system is then designed and implemented from scratch. A few technologies have already been researched by the industry mentor as well as academic supervisor. A detailed analysis of these is done during the course of this project.

1.2 The Company

MCS was founded in 2006 to develop new ways for humans to easily interact with their remote assets and information relating to important assets via mobile phone. MCS is now engaged in developing products in the marine asset monitoring, mobile commerce, and remote control and monitoring sectors. The focus is on developing new and easier ways for humans to use their mobile phones/internet to remotely control, monitor and interact with remote assets, information and services.

MCS major focus is the human interface to mobile devices. This involves optimizing informatics and human interface to mobile devices by the innovative use of application software in mobile phone and related devices.

2 Project Details

This section will present the problem that this project solves and the aim of this project. Also, it will show a basic workflow of projects at MCS.

2.1 Motivation

The motivation of this project is to deliver a system which provides a virtual presence, with an asset of value, to the user. This system should not only be able to monitor the asset but also deliver alert messages to the user and also the ability to control this asset remotely.

A server that serves information for an organization is an asset of value to that organization. When a server breaks down, the cost of not providing a service is very high. A service needs to be continually served by its server. For e.g., if a web-site hosting server breaks down, the number of hits that the web-site gets goes down. This results into loss of advertising and/or poor quality of service. If a server of a banking system breaks down, many services provided by the bank will stop. There are different ways to tackle this problem. One of the ways would be to have a backup server. Another would be to setup grid computing (Snavely, et al., 2003). But faster the breakdown is detected and faster the user is informed about the breakdown, faster the user can act upon the situation. And hence reducing the time taken to get the server up and running. When monitoring a server, a user should be able to monitor it from anywhere, whether he is on a computer or not.

2.2 Goals

The aim of this project is to monitor and control a Windows based server running on .NET platform, remotely. To monitor a server remotely, this information has to be sent over an easily accessible medium - Internet. This enables us to monitor it using a browser from a laptop or a computer. Internet also makes it possible to view this information on a mobile phone. Mobile phones enable the user to monitor the server on-the-go. They can access this information from anywhere, anytime. This system should be able to work on a very wide range of mobile devices.

The system should be able to control facilitate user to control a few basic operations of the server. Below is the list of elements that this system is supposed to monitor. Also, the basic operations that are to be performed on the server are listed below.

The system should be able to monitor the following elements of the server:

- ✓ Average CPU utilization, an average CPU utilization metric must be maintained (resettable).
- ✓ Average CPU temperature, an average CPU temperature must be maintained (resettable).
- ✓ Peak CPU temperature, a peak CPU temperature must be maintained (resettable).
- ✓ Average RAM usage, an average RAM utilization metric must be maintained (sample time base must be configurable).
- ✓ Peak RAM usage, a peak RAM usage metric must be maintained (resettable).
- ✓ Current hard drive usage, the current value of used hard drive space (all installed hard drives).
- ✓ (S.M.A.R.T) Average hard drive temperature, an average hard drive temperature must be maintained (resettable).
- ✓ (S.M.A.R.T) Peak hard drive temperature, a peak hard drive temperature must be maintained (resettable).
- ✓ (S.M.A.R.T) Error read rate, the error read rate will enable the diagnosis of a failing hard drive (research to be done into what suitable information should be retrieved from the S.M.A.R.T information to diagnose a failing hard drive).
- ✓ List current executing processes, a snap shot of the names of the systems currently executing processes.
- ✓ A break down of each executing processes resource utilization (CPU usage, RAM usage etc).

The system should be able to perform the following basic operations on the server:

- ✓ Reset the server.
- ✓ Start a service.
- ✓ Stop a service.
- ✓ Start a process.
- ✓ Stop a process.

The above functionalities should be performed by a password protected application (web or client). Along with a mobile application, the system should also provide these functionalities of the server through a website. The system should also be able to send alerts to the user through SMS in case of any problems detected in the server. The diagram below shows an overview of the system.

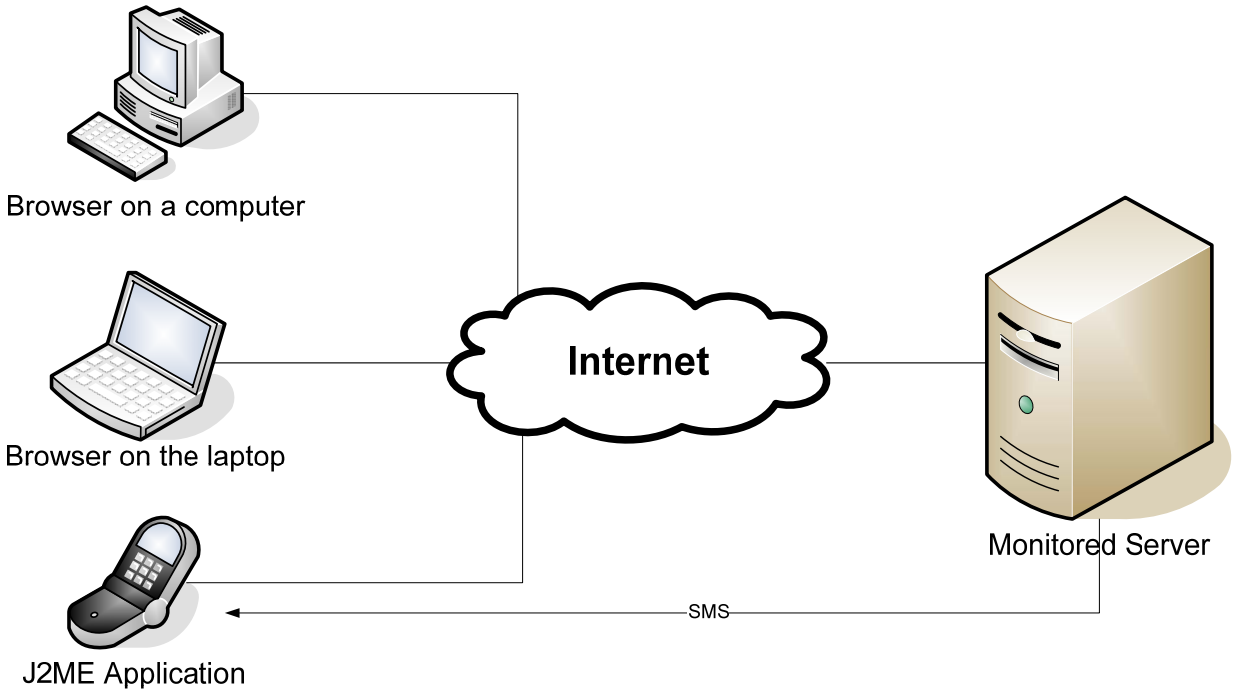

Figure 1: Overview of the system.

2.3 Learning Opportunities

This project will give me a lot of learning opportunities starting from project basic research of technologies to project management. Since the system is to be build from scratch and I am the only person working on this project (of course under supervision of my academic supervisor and academic mentor), there is quite some project management tasks to be learned. Another thing to learn here is how to document the project details and how important the communication between the developer and the client/company is.

3 Technical investigation

This section is divided into two sections. First section discusses the ways in which the elements of the server can be monitored and controlled. Second section talks about the ways in which the mobile application can be implemented. The section after that would introduce us to other key technologies.

3.1 Monitoring Server

For a .NET platform, the best way to get information about the hardware, services running on the computer and processes running on the computer is by using Windows Management Instrumentation (WMI). Here, we will investigate what WMI is and how useful it is to fulfill our goals of the project.

3.1.1 Windows Management Instrumentation (WMI)

"WMI is an infrastructure for managing data and operations on Windows Operating systems." (Windows Management Instrumentation (WMI), n.d.). WMI is used to automate administrative tasks on remote computers. It can also supply management data to different parts of an operating system. "WMI uses the Common Information Model (CIM) industry standard to represent systems, applications, networks, devices, and other managed components." (About WMI, n.d.). These WMI applications can get management data or perform operations on the operating system or its services on a variety of languages. We can write a client script or application to get data about hardware or software from WMI. Also, we need to provide data from hardware or software to WMI by creating a WMI provider. Provider basically translates the WMI script into its corresponding windows API. It gets the information and then provides that information to the client (The .NET Show: WMI Scripting, 2006). All services, processes, hardware such as monitor, hard drive, memory (RAM), etc. have their own WMI class. These classes are the providers of management data. WMI basically provides a level of abstraction to get specific information about the operating system and the processes and services on it as well as perform operations on them. This achieved by using WMI queries. This query language is called WQL (WMI query language) (Querying with WQL n.d.). WQL is a subset of ANSI SQL. Below is a snippet to give an example of a WQL query.

SELECT FreeSpace FROM Win32_LogicalDisk WHERE DriveType=3

The above query will return the free space in a disk where the type of disk is 3. Similarly,

SELECT CurrentClockSpeed FROM Win32_Processor WHERE Manufacturer='Intel'

will get the current speed of all the processor whose manufacturer is Intel. This makes it easy to retrieve information about the system. In a similar fashion we can also query information about processes and services using Win32_Process and Win32_Service classes.

3.1.2 WMI .net

The Technology Summary for WMI .NET link from (WMI .NET Overview, n.d.) says "Windows Management Instrumentation (WMI) is a component of the Windows operating system that allows you to programmatically access management information in an enterprise environment". The WMI .NET framework is built on the same WMI technology. It gives us the benefit of using WMI technology in a .NET application. Using the System.Management namespace in the .NET framework, we can run WMI on different .NET languages like C# .NET, Visual Basic .NET, etc. System.Management.

Instrumentation workspace can be used to instrument the application so that it can provide information to WMI layer about the behavior of the application. The key benefits of WMI are as follows (Benefits of WMI in .NET Framework, n.d.):

- ✓ Leverage of common language runtime features, such as garbage collection, custom indexer, and dictionaries.
- ✓ Definition of classes and publication of instances entirely with .NET Framework classes to instrument applications so the applications can provide data to WMI.
- ✓ Simplicity of use.
- ✓ Access to all WMI data.

There are a few limitations of WMI which are not relevant to our requirements. Hence, I have not placed them here. Below is a diagram showing the framework of WMI.

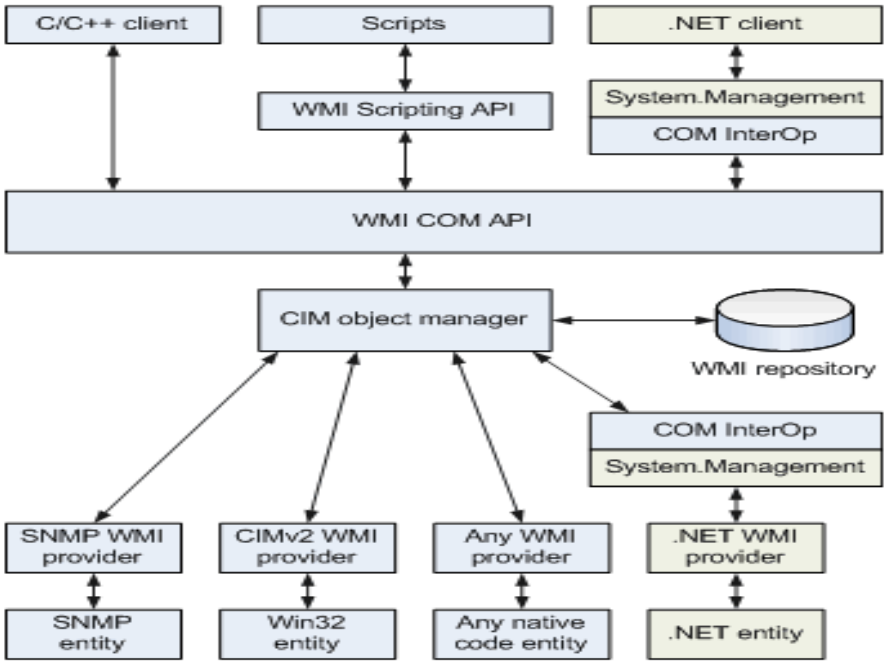

Figure 2: Overview of WMI framework.

As seen in the above figure, the WMI COM API acts as an interface between different types of manage objects like Win32 entities, a .NET entity, etc. And any programming languages, including scripting languages can access this information via WMI COM layer.

3.2 Mobile Application

The purpose of the mobile application is to be able to display the information about the elements of the server on a mobile device. Many different types of technologies are present to make a mobile application. A few of these are discussed below.

3.2.1 Java ME

In (White, 2001) James White shows the very basic understanding of the Java ME platform. Java ME stands for Java Micro Edition. It's a lighter version of java which can be used on a low power, network connectivity and graphical user interface capabilities (White, 2001). Java ME provides an application

platform to a very wide range of devices. This is because it provides the ability of writing the code once and running it on several platforms.

Java ME is divided into three elements: configurations, profile and optional APIs.

3.2.1.1 Configurations

Specifications that address the virtual machine and general Java API running on a large group of devices are called configurations. Java ME has been divided into two basic configurations, one targeted to the small devices and the other towards more capable mobile devices. The configuration for the small devices is called “Connected Limited Device Configuration” (CLDC) and the other is called “Connected Device Configuration” (CDC). CDLC is specifically designed to meet the needs for a Java platform to run on devices with limited memory, processing power and graphical capabilities.

3.2.1.2 Profiles

On top of these on configurations, it also specifies a number of profiles for describing a set of higher-level APIs that could further define the application. A profile addresses the needs of the specific subset of devices.

MIDP 1.0

MIDP stands for Mobile Information Device Profile. The MIDP profile the core profile for CLDC. It was developed specifically for mobiles and pagers (Kolsi, 2004). A MIDP application is called a MIDlet. A MIDlet has no knowledge of other MIDlets through the MIDlet API. MIDP 1.0 environment includes APIs that can define the MIDP application model and capabilities available for MIDlet applications.

MIDP 2.0

“MIDP 2.0 enhances the overall end user experience, improves application portability, and provides greater extensibility” (What's New in MIDP 2.0?, n.d.). MIDP 2.0 has more flexibility layout for greater

application portability. Its also has greater extensibility. For e.g., it has something called Custom Items. This is used mainly to create our own forms and UI components. MIDP 2.0 also has support for media files. (What's New in MIDP 2.0?, n.d.) also describes about the new OTA (Over-the-air) feature which only a recommended feature in the previous version. The MIDP specifications define how the application suites would behave in a particular environment.

Security comparison

(Kolsi, 2004) talks about a lot of security issues concerning Mobile Environment including threats in this article. They talk majorly about Internet environment threat, Mobile Network threats, Java threats and MIDP 1.0 security problems. This article then evaluates the problems MIDP 1.0 security. It evaluates the security threats in the MIDP 1.0 and tells the solutions that are provided in MIDP 2.0. The table below summarizes the problems faced with MIDP 1.0 and the solutions provided by MIDP 2.0.

Shown below is the table that describes the level of protection provided by MIDP 1.0 and how the threat not handled in MIDP 1.0 is handled in MIDP 2.0.

Threat	*MIDP 1.0*	*MIDP 2.0*
Network Security threats in GSM/GPRS or internet protocols	Application level protection on top of HTTP	**SSL/HTTP security protocols**
Unauthorized network connections	User permissions	**Protection domains, security policy and user permissions.**
Physical threats against devices	Application level protection for data	**Application level protection for data.**
Malicious applications performing Dos	Flawless JVM and sandbox	**Flawless JVM and sandbox, signed applications and authentication of origin.**
Application binary integrity		**Signed applications**

Table 1: Security comparison between MIDP 1.0 and MIDP 2.0

In (Debbabi et. al., 2005), Mourad Debbabi et al. describe the wide spread use of Java ME and explain how it is susceptible to security threats. They then talk about the CLDC security architecture and describe how the CLDC architecture affect the security of a mobile application when it is ran on MIDP 1.0 environment. They then talk about the security problem solved by the MIDP 2.0 security model.

3.2.1.3 GUI

Java ME has a very limited support for GUI. The MIDP 2.0 profile features very basic GUI components. A very good looking front-end application is very necessary for the project. This makes the product of this project even more competitive in the market. As recommended in (Wesson, 2005), the users are very much inclined towards good visual appearance and screen layouts. MIDP clearly lacks these. So we need to find out different ways to enhance the appearance of the mobile application. There are a few commercial as well as Open Source GUI APIs available in the market.

Polish

J2ME Polish is a commercial tool for building GUI for J2ME applications. J2ME Polish is suite of tools for creating "polished" J2ME applications. Each tool meets a definite need of J2ME developers: Build-tools with an integrated device-database, a powerful GUI, a framework for building localized applications, a game-engine, a logging framework and a collection of utilities. It supports a very wide range of mobile devices as well. J2ME Polish also has an Open Source version which is available under the GNU GPL license. This license needs the application made using J2ME Polish to be open source as well. If the application developing company wants it to be a closed source application, then the company has to pay for the licenses. The cost of licensing this application is very high. One of the future works could be to extend the application built in this project using J2ME Polish.

For the mobile application in this project, the decision was made to use Java ME because it was supported on a very wide range of devices. Other technologies such as FlashLite, etc. are platform dependent and are not supported on a very high range of devices. Hence, it was decided to use Java ME. Also, it was decided to create our own API for controls to be used on the mobile phone. This would give other developers at MCS a chance to aid GUI development of other projects as well as get some help

from other projects to develop their own GUI. Also, browser applications (thin client) for mobile application would increase the data transfer because the HTTP header for browser pages is much higher compared to that for mobile application such as Java ME.

3.3 Embedded Database

(Koopmann, 2005) An embedded database is a software component that is a part of an application and not a separate running application. The end user has no knowledge of its existence. Embedded databases are fast and reliable. In this project, an embedded database can be used to store the user information for authentication and for logging the report generated by the system.

3.4 Isolated storage

Isolated storage is, essentially, a special spot on the hard drive that only your application can find. The .NET Framework takes care of managing the actual disk storage. Once you've opened your isolated storage, you can create files or directories in it, and treat it pretty much like any other disk space. The nice thing is that even if your application doesn't have permissions to access the file system, you can still use isolated storage.

4 Design

Here, we will see the design of the server part of the system that queries the operating system and gets information needed to monitor the server. Below is the architecture diagram of the system that would reside on the server side.

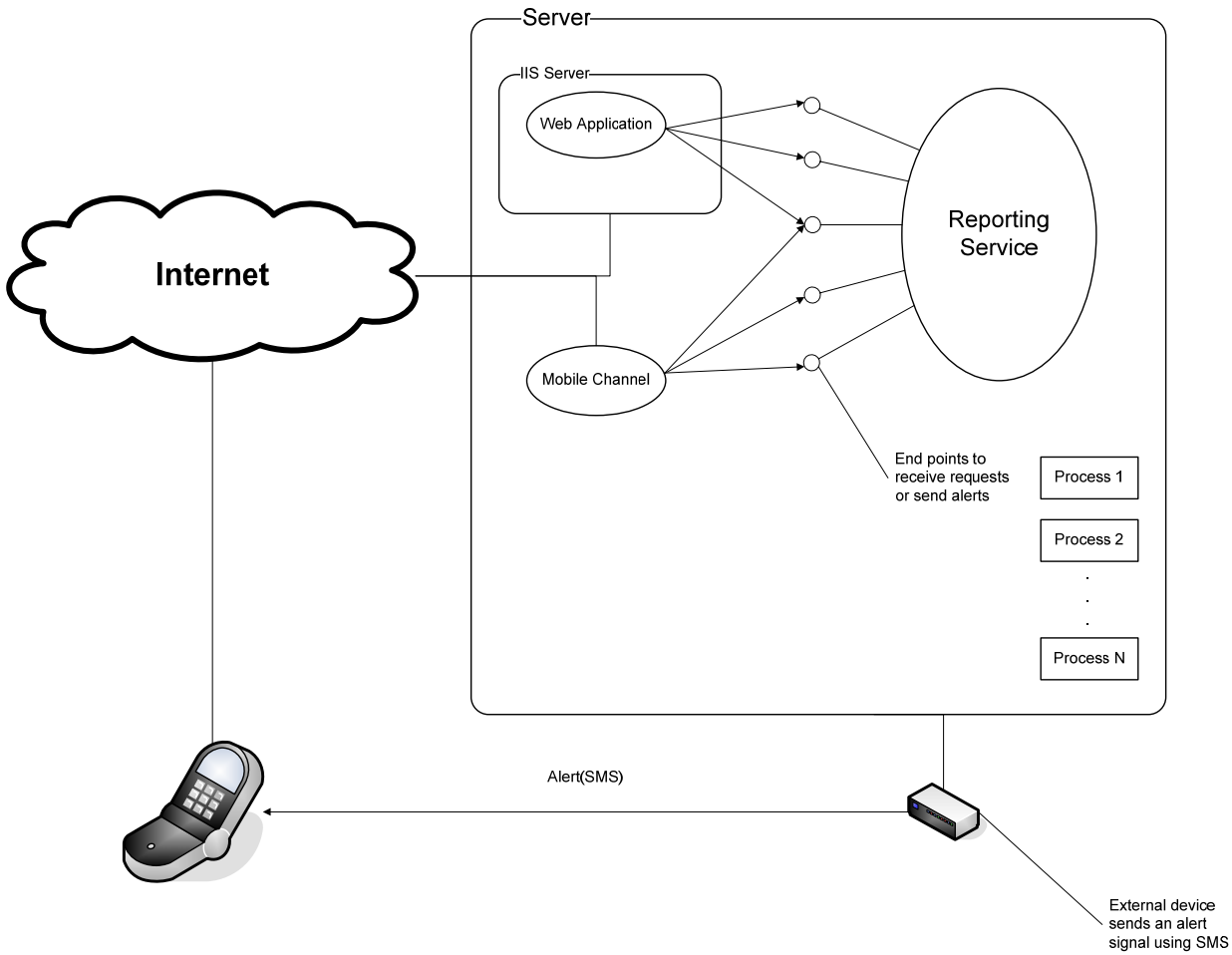

Figure 3: Architecture – Server-Side.

The above diagram shows the basic architecture of the part of the system that would reside on the server side. The different components are explained below:

4.1 Reporting Service

Reporting service is the part of that system that actually monitors the system health. It checks for the health of the system at regular intervals and makes this information organized for other parts of the system. This interval is configurable by the user. The reporting service will build using the .NET 2.0 framework. The programming language used will be C#. This reporting service is made up of three different components: agent, agent environment and reporting service provider.

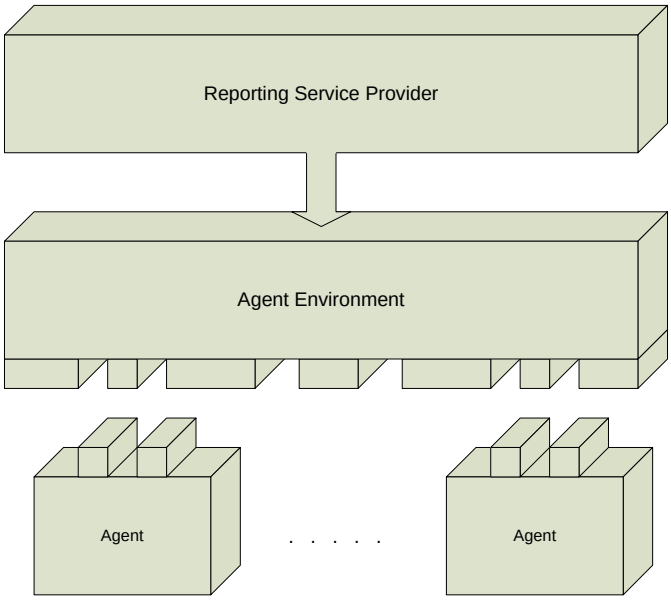

Figure 4: Architecture – Reporting Service

4.1.1 Agent

Agent is a component that monitors single element of the server. For e.g., a Processor agent will monitor only the CPU. Disk agent will only monitor a disk. This component uses WMI .NET to get information from the operating System. This component will also log the information that is retrieved using WMI for back tracing the fault.

4.1.2 Agent Environment

Agent information is a component of the system that gathers data from all the agents and makes it available for other components. The agent environment provides data according to the data requested by other components. For e.g., if a component requests information regarding the CPU, the agent environment will get the relevant data from the Processor agent and pass it back to that component.

4.1.3 Reporting Service Provider

A reporting service is a component which will gather information from the agent environment and make it available for other parts of the system to show these to the user. The end-points denote that this component will be listening for requests and provide the appropriate data back to the requesting component. This reporting service will also check for authentication. This will be done using an embedded database (Koopmann, 2005).

4.2 Web Application

A web application will run on an IIS server on the server machine. This web application will be able to authenticate users into the system. It will enable users to monitor all the relevant elements about the server hardware, processes and services that are running on the server. It will have a good user interface and be easy to use. Users will be created by MCS as per the licensing policy. The web application will be created using ASP .NET 2.0.

4.3 Mobile Application

A Mobile Application will enable user to access the information regarding the elements of the server on a mobile phone. This application will be build using Java ME. This application will accept xml (Quin, 2007) data from the server and display this information on the mobile application. The user interface

will be semi-dynamic (depended on the type of information displayed). The user will need to authenticate himself before requesting any data from the server.

4.4 Mobile Channel

A mobile channel will facilitate the HTTP data to be transferred between the reporting service and the mobile application. Having a separate mobile channel has a few advantages like the mobile channel can send data with minimal HTTP headers. Custom encoding and compression can be performed for the mobile application. Upon receiving a request, this channel will query and retrieve data from the reporting service. This data will then be transferred into an xml string and passed to the mobile application by attaching required HTTP headers. This channel will act as a proxy for the mobile application. This will be built using .Net 2.0 framework in C#.

4.5 External Alerting Device

An external alerting device will be used to alert the user in case of an emergency such as system shutdown or abnormal behavior of the server resources. This device will be a cell phone module connected to a TINI device. The TINI device will be connected to the server through serial port. It will send continuous ping requests to the server. Upon not receiving a reply from the server, the cell modem will send an alert signal to the user's mobile phone via SMS.

Modularity of components allows for extensibility and reuse of these components. This entire system will be built on Component Based Software Engineering principles. The components will interact with each other using their interfaces. This makes the system re-usable and easy to manage and debug.

5 Next Step

The next step is to finish the implementation of the reporting service and start building the web application and the mobile channel. The text cases for each module of the system are built on-the-go for every part of the system. This will save time in building and running test cases at the end of the project. The project will be well documented and hopefully follow the road map. There are a few more technologies to be looked at. E.g., AJAX for implementing the web application. The last part would involve building the Java ME mobile application. This will be developed using custom GUI functions.

As a future work, I would like to include that this system can be extended to Linux servers using a technology called Mono. Mono is used to run .Net applications on the Linux operating system. Currently, Mono does not support the System.Management package. This system can be extended when System.Management package is added to the Mono project.

6 Conclusion

The project has so far been into its design stages. The project has been going at a slower pace then estimated. Hopefully the project will be back on track in late June. The design and implementation of the project is expected to finish by the end of September. As per the plan, the testing phase has been allocated a month. This will allow changes to be made to the system at later stages if something goes wrong. During the 2nd part of this project, I have planned to increase my time commitment towards the project.

I have gained a lot of knowledge about project management till now and I am sure that I will learn a lot till the last day of the project. This project has taught me the value of documenting, planning and scheduling a project and also a few concepts about project management. Also, I have developed my research skills in the course of this project and learned the value of communication. I have learnt many technologies like Java ME, Managed code, WMI and a few related .NET technologies. In future I hope to develop my ASP .NET as well as XML skill in the due course of the project.

Bibliography

About WMI. (n.d.). Retrieved March 2007, from MSDN: http://msdn2.microsoft.com/en-us/library/aa384642.aspx

Benefits of WMI in .NET Framework. (n.d.). Retrieved March 2007, from MSDN: http://msdn2.microsoft.com/en-us/library/ms186144.aspx

Debbabi, M., Saleh, M., Talhi, C., & Zhioua, S. (2005). Security analysis of mobile Java. *Database and Expert Systems Applications, 2005. Proceedings. Sixteenth International Workshop* (pp. 231- 235). IEEE CNF.

Distributed computing. (n.d.). Retrieved June 1, 2007, from Wikipedia, the free encyclopedia: http://en.wikipedia.org/wiki/Distributed_system

Feinstein, W. P. (2000). A study of technologies for Client/Server applications. *Proceedings of the 38th annual on Southeast regional conference* (pp. 184 - 193). Clemson, South Carolina: ACM Press.

Kolsi, O. (2004). MIDP 2.0 security enhancements. *Proceedings of the 37th Annual Hawaii International Conference* (p. 8 pp.). IEEE CNF.

Koopmann, J. F. (2005, April 12th). *Embedded Database Primer*. Retrieved from DBAzine: http://www.dbazine.com/ofinterest/oi-articles/koopmann5

Querying with WQL. (n.d.). Retrieved March 2007, from MSDN: http://msdn2.microsoft.com/en-us/library/aa392902.aspx

Quin, L. (2007, May 8th). *Extensible Markup Language (XML)*. Retrieved from World Wide Web Consortium: http://www.w3.org/XML/

Snavely, A., Chun, G., Casanova, H., Van der Wijngaart, R. F., & Frumkin, M. A. (2003). Benchmarks for grid computing: a review of ongoing efforts and future directions. *ACM SIGMETRICS Performance Evaluation Review. Volume 30*, pp. 27 - 32. ACM Press.

The .NET Show: WMI Scripting. (2006, April). Retrieved from MSDN: http://msdn.microsoft.com/theshow/episode.aspx?xml=theshow/en/episode055/manifest.xml

Using WMI. (n.d.). Retrieved March 2007, from MSDN: http://msdn2.microsoft.com/en-us/library/aa393964.aspx

Wesson, J. v. (2005). mplementing mobile services: does the platform really make a difference? *Proceedings of the 2005 annual research conference of the South African institute of computer scientists*

and information technologists on IT research in developing countries. ACM International Conference Proceeding Series; Vol. 150, pp. 208 - 216. White River, South Africa: South African Institute for Computer Scientists and Information Technologists.

What's New in MIDP 2.0? (n.d.). Retrieved March 2007, from Sun Developer Network: http://java.sun.com/products/midp/whatsnew.html

White, J. (2001). An introduction to Java 2 micro edition (J2ME); Java in small things. *Proceedings of the 23rd International Conference on Software Engineering* (pp. 724 - 725). Toronto, Ontario, Canada: IEEE Computer Society.

Windows Management Instrumentation (WMI). (n.d.). Retrieved March 2007, from MSDN: http://msdn2.microsoft.com/en-us/library/aa394582.aspx

WMI .NET Overview. (n.d.). Retrieved March 2007, from MSDN: http://msdn2.microsoft.com/en-us/library/ms257340.aspx