Design of a WBEM-based Management System for Ubiquitous Computing Servers

So-Jung Lee[1], Mi-Jung Choi[1], Sun-Mi Yoo[1], James W. Hong[1]
Hee-Nam Cho[2], Chang-Won Ahn[2], Sung-In Jung[2]
[1]Dept. of Computer Science and Engineering, POSTECH
[2]Digital Home Research Division, ETRI
{annie, mjchoi, sunny81, jwkhong}@postech.ac.kr, {hncho, ahn, sijung}@etri.re.kr

Abstract

As the Internet evolves and wireless network technologies develop, people's need to get a network service at anytime from anywhere is growing, and the age of ubiquitous computing is coming in earnest. A ubiquitous computing server receives and processes lots of information from various sensors in a ubiquitous computing environment and plays a role in providing various useful services. In this paper, we analyze the requirements for managing the ubiquitous computing servers based on WBEM technologies which are being standardized in DMTF. We present the design of our management system, which satisfies the requirements. We also examine and compare available WBEM implementations by performing benchmarking testing. Based on the benchmarking results, we present our choice of WBEM implementation, which is being used for implementing our ubiquitous computing server management system.

1. Introduction

As the Internet continues to grow and the wired/wireless network evolves, users want to get various services at anytime and anywhere by connecting to the network. The idea used to realize this service is ubiquitous computing. Ubiquitous computing is a new paradigm for distributed interactive systems, which moves computers into the background of people's attention while using them to support their activities and interactions in the workplace and beyond. The ubiquitous computing paradigm is underpinned by the development of devices small enough to be embedded in almost anything, of networks that provide a dense inter-connection of a very large number of components, and of sensing technologies that enable systems to become aware of their physical environment.

A ubiquitous computing server is required to act as an intermediate to provide a ubiquitous computing service that can be used anytime and anywhere. Therefore, a ubiquitous computing server (or U-server) also needs to be connected to the network and needs to be managed. A standard management mechanism is required to manage not only the U-server but also the services that are offered through the U-server.

The Distributed Management Task Force (DMTF) [1] working group is standardizing the Web-Based Enterprise Management (WBEM) [2] architecture, which is using the Common Information Model (CIM) [3] for information modeling, Extensible Markup Language (XML) [4] for management information encoding and HTTP for the transport protocol. In this paper, we analyze the requirements for managing the ubiquitous computing servers. We present the design of our management system that satisfies the requirements. We also examine and compare available WBEM implementations by performing benchmarking testing. Based on the benchmarking results, we present our choice of WBEM implementation, which is being used for implementing our ubiquitous computing server management system.

The remainder of this paper is organized as follows. Section 2 presents the general overview of ubiquitous computing, and WBEM. Section 3 discusses the requirements for managing U-servers. Section 4 presents the design of server management system for the U-servers. Section 5 briefly examines few freely and commercially available WBEM implementations and compares them. Finally, we summarize our work and discuss plans in Section 6.

2. Related Work

In this section, we first introduce the general concept of ubiquitous computing and the functions of a U-server. Then we give an overview of WBEM.

2.1 Ubiquitous Computing

Ubiquitous is a Latin word whose meaning is ‘being or seeming to be everywhere at the same time’. Ubiquitous computing means a user can connect to the network and be serviced regardless of time or location. To compose a ubiquitous network, every computer has to be connected to it. In a ubiquitous computing environment, components like sensors, drivers or processors can be embedded into daily items, which would add new services such as information processing and exchanging to their original ability. The information can be anything which exists around them and a high quality service such as a location-based service, an automated merchandise management service, an automated medical service or a disaster prevention service can be provided by collecting this information.

Figure 1 illustrates a ubiquitous computing environment. U-servers are connected to the Internet and are distributed throughout the ubiquitous computing environment. U-servers collect the data from the sensors that are embedded in the things like home PC, game machine, electric home appliance, mobile phone or multimedia kiosk and they are also connected to the Internet. U-servers provide the service to other items according to the data it collects. Specifically, U-server’s function is to extract the required information from the various sensors and forwards it to the related applications. For example, to provide the IP-based storage service, U-servers stores the data that we work at home, and presents it at the office’s PC without any help of portable storage devices.

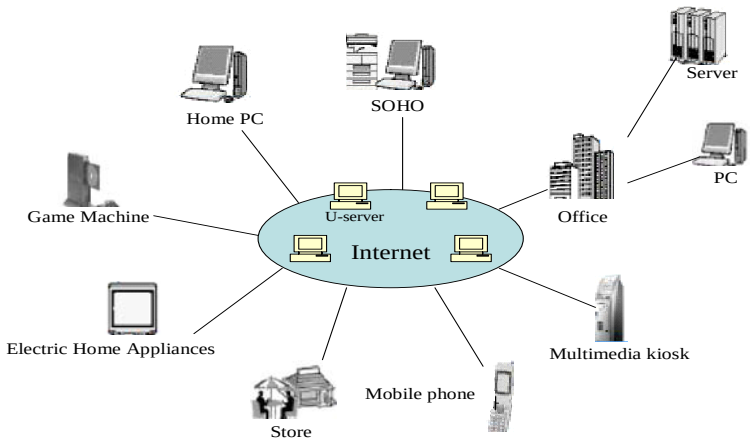

Figure 1. Ubiquitous Computing Environment

2.2 Overview of WBEM

WBEM is an initiative of DMTF and it includes a set of technologies that enables the interoperable management of an enterprise network. The DMTF has developed a core set of standards that make up WBEM, which includes a data model, the CIM standard; an encoding specification, CIM-XML encoding specification; and a transport mechanism, CIM operations over HTTP.

The CIM specification is the language and methodology for describing management data. CIM is an object-oriented schema for modeling the objects. The CIM schema can be divided to three areas; the core model, the common model and the extension model. First, the core model captures notions that are applicable to all areas of management. Second, the common model is an information model that captures notions that are common to a particular technology. For example, it includes the model for systems, applications, networks and devices. Last, the extension model represents technology-specific extensions of common models.

The CIM-XML encoding specification defines XML elements, written in Document Type Definition (DTD) that can be used to represent CIM classes and instances. The CIM operations over HTTP specification defines a mapping of CIM operations onto HTTP that allows implementations of CIM to interoperate in an open, standardized manner and completes the technologies that support WBEM. Therefore, in the WBEM architecture, the management information is described by the CIM schema, converted to XML, and finally embedded in an HTTP payload to transport to the target node.

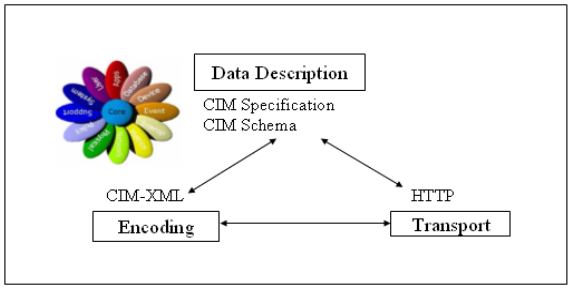

Figure 2. Relationships among WBEM Standard Technologies

Figure 3 illustrates the WBEM architecture, which includes a *WBEM Client*, and *WBEM Server*. *WBEM Server* has *CIM Object Manager* (*CIMOM*) which is a central component that routes information about objects and events between components. The *CIMOM* responds to the operations defined in the "CIM operations" specification such as create, modify, and delete. It also checks the syntax and semantic of the messages, and provides security. Providers are so-called instrumentation agents. Namely, *Providers* actually obtain the information from the resources and forward it to the *CIMOM*. A *WBEM Client* is commonly represented as the management application, and it can get the information by sending a request message to the CIMOM instead of directly accessing the providers.

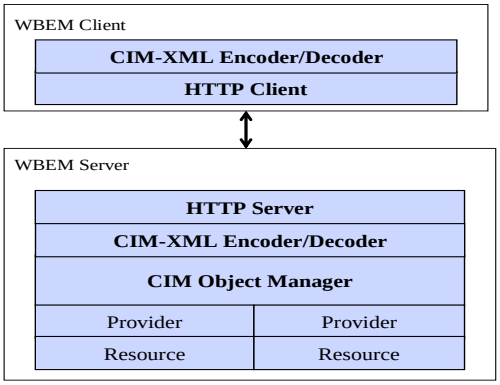

Figure 3. WBEM Architecture

3. Requirements

In this section, we analyze the non-functional and functional requirements for managing U-servers. The management system for U-servers is composed of a manager and an agent. A manager is a WBEM client and an agent is a WBEM server embedded in the U-server being managed.

3.1 Non-functional Requirements

The non-functional requirements for managing U-servers are as follows.

(1) A manager and an agent must be designed flexibly, and they must have modularity in order to support any architecture, such as a centralized or distributed architecture, so that they can manage multiple U-servers efficiently.

(2) Various kinds of services are offered through U-servers, and U-servers communicate with heterogeneous application servers that are distributed in the ubiquitous computing environment. Several management operations are necessary to manage U-Servers communicating these application servers. Therefore, a standard management protocol with abundant management operations is required. We follow the WBEM standard architecture which is a DMTF server management technology to fulfill this requirement. That is, we use the CIM meta-schema to define the management information, and we use the XML/HTTP protocol to exchange the

management operations.

3.2 Functional Requirements

The following are the functional requirements of the management system for managing U-servers.

(1) The management information is described by the CIM schema, and after being encoded in an XML message, it is sent by being embedded in an HTTP message payload. The manager and agent have to support the information modeling mechanism and include the required modules to process the management operations according to the WBEM standard.

(2) A Web-based user interface is necessary for the administrators to connect to the management system at anytime from anywhere. This requirement can be easily satisfied using the WBEM technologies.

(3) A manager has to provide topological management functions that can register, modify or delete agents. It has to offer the functionality to monitor and set-up the management information about the U-servers. It also has to provide an analyzing function that can store the necessary information in a database and analyze it.

(4) The agents may include multiple providers to manage each resource. Multiple providers interact with the CIMOM through a provider manager, and each provider needs to offer a backend interface for each resource. An agent has to supply system basic information and system performance information to the manager.

(5) An agent has to provide a mechanism initiating the connection for sending the notification information to the subscribed manager in the case that an error is generated.

(6) A U-server has to manage the list of users who register to obtain a service as well as a list of the services to be provided. It also has to provide application services in response to a user's request.

(7) A U-server has to guarantee the quality and the availability of the services to the users. That is, a U-server has to provide a monitoring and control function for QoS to guarantee the reliability of the application service in the ubiquitous computing environment.

(8) An agent has to support a remote provider for small devices as well. Methods to input and output the management information on behalf of small devices are required. This is because a WBEM server is not installable in most small devices due to limited computing resources. Generally, a WBEM server requires a lot of system resources. Therefore it is irrational to embed a CIMOM instance in all devices.

(9) A repository technology has to be provided to minimize the system resources used by a WBEM server that is embedded in an agent.

(10) An agent has to provide an integration function for managing devices embedding SNMP agent. Currently, most network devices on the Internet and enterprise networks use SNMP for the management protocol. Network device vendors do not want to adopt a new management

technology without supporting a previous management paradigm. Therefore, integration with SNMP-enabled devices must be provided.

(11) The management system has to provide a security mechanism because it connects to the U-servers and plays an important role in the server setting. Therefore it needs to permit access only to authenticated administrators. Also, the management information has to be protected.

4. Design

In this section, we present the design of a U-server management system based on the WBEM technologies and which satisfies the requirements mentioned in Section 3. First, we present the management information that is defined by the CIM core, common and extension schemas. Second, we present the management protocol to which the CIM operation over HTTP mechanism is applied. Finally, we describe the management system architecture designed for managing U-servers.

4.1 Management Information

We can sufficiently describe the management information for the ubiquitous computing server with DMTF's CIM core and common models. The management objects can be described in XML as well as the Management Object Format (MOF) [5].

The CIM schemas used in the management system for the U-servers are the CIM application, CIM database, CIM device, CIM metrics, CIM network and CIM user, all from the common model. The properties and operations of the services provided by the U-server can be described through the *Application Service Class* included in the CIM application model schema. The CIM database schema can describe the information which the U-server collected from each sensor and stored in the database. The CIM device schema covers the low-level devices such as sensors, batteries and fans of a U-server as well as high-level abstractions such as storage volumes of a U-server. The CIM metrics schema can include the metric information such as the processing or the resource utilization information of a U-server in order to guarantee QoS. The CIM network model is required to describe network information such as the network connectivity state, because U-servers are connected in a network by wire or wireless. The CIM user schema is needed for a U-server to provide security services such as user authentication and access control. For example, the authentication service class in the CIM user model verifies a user ID and password, and it controls access to the resources.

4.2 Management Protocol

For the management protocol, we use CIM to XML mapping [6] and CIM operations over HTTP [7], both of which are being standardized by DMTF. The specification of the CIM to XML mapping defines the XML schema used to describe the CIM object in XML for encapsulation over HTTP. Both CIM classes and instances must be valid XML documents for this protocol.

CIM operations over HTTP allow implementations of CIM to operate in an open, standard manner. It describes how CIM operations are encoded in the HTTP payload using XML and defines the syntax and semantics of the requests and their corresponding responses. For basic information gathering functions we use the GetClass, EnumerateClasses, EnumerateClassnames, GetInstance, EnumerateInstances, EnumerateInstanceNames and GetProperty operations. For the basic writing function we use the SetProperty operation. For schema or instance manipulation we use the CreateClass, ModifyClass, DeleteClass, CreateInstance, ModifyInstance, and DeleteInstance operations. For association traversal operations we use the Associators, Associators Names, References and ReferencesName operations. For query operations we use the ExecQuery operation. Finally, for the qualifier declaration, we use the GetQualifier, SetQualifier, DeleteQualifier and EnumerateQualifier operations. We use HTTP over SSL (HTTPS) [8] for secure communication.

4.3 Management Architecture

In this section, we present our management architecture. We first describe the overall management architecture. We then describe the manager and agent architectures.

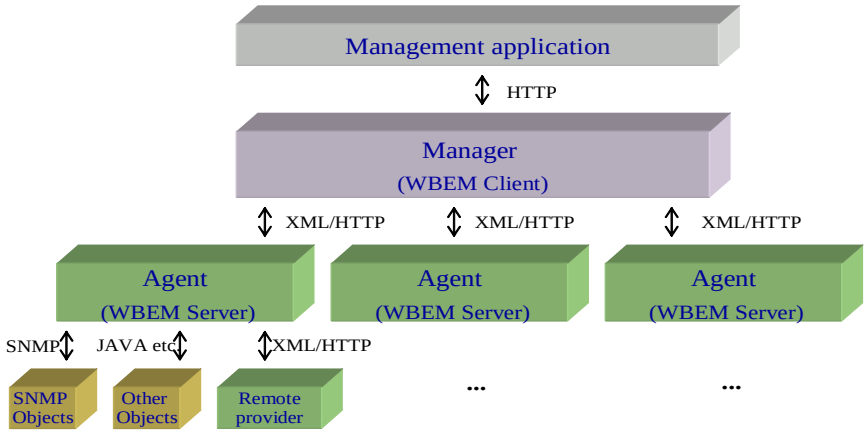

Figure 4. Overall Management System Architecture

Figure 4 illustrates the overall architecture of the server management framework we have designed. The management system is based on the traditional manager-agent architecture. In the viewpoint of a WBEM-based architecture, a manager is a *WBEM Client* and agents are the *WBEM Servers*. Users can access a manager through the management application, and the protocol between the manager and the agents is XML/HTTP to which the CIM operations over HTTP mechanism is applied. An agent can get the information from an SNMP object, Java object or any other objects. An agent can access resources with a resource-specific protocol such as SNMP, or it can get the information from a remote provider that is installed in a remote device. The remote provider sends the local information to the agent. A more detailed explanation of the remote provider is discussed in the agent architecture.

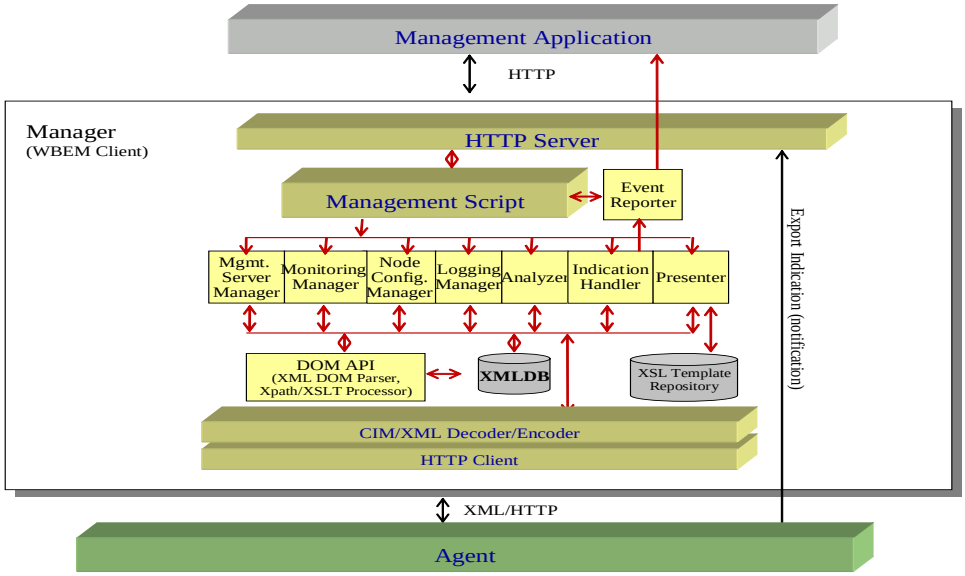

Figure 5. Manager Architecture

Figure5 illustrates the architecture of a manager. An *HTTP Server* is used to provide administrators with a Web-based user interface and to receive requests from the management application and pass them to the management components through the *Management Script*. Also, an *HTTP Server* is used to receive the notification information exported from an agent. The *CIM/XML Decoder/Encoder* converts the message to follow the WBEM standard, and the *HTTP Client* plays a role in the interface module of the device that exchanges the management information with an agent. The *XMLDB* is used to store the management information for long-term analysis. The *XSL Template Repository* stores XSL [9] files for generating HTML documents from XML documents. The *DOM API* [10] is used to implement the management functions because the management information is represented in XML data.

The major components of a manager are the *Management Server Manager*, the *Monitoring Manager*, the *Analyzer*, the *Notification Handler*, the *Logging Manager*, the *Presenter* and the *Event Reporter*. The *Management Server Manager* manages the configuration of the management processing environment and handles the topologies of a device and group tree structure. This component also manages an administrator account list. The *Node Configuration Manager* is a module to get and set the configuration of a managed node. The *Monitoring Manager* is a module to obtain device monitoring information, such as node status and in/out traffic. The *Logging Manager* logs the necessary data in the DB and analyzes it per the administrator's request. The *Indication Handler* receives indication from the managed nodes, stores them in the DB tables and sends a meaningful indication to the *Event Reporter*. The *Event Reporter* generates appropriate events and sends them to administrators via email, pager, fax or SMS. The *Analyzer* analyzes the collected management information and provides meaningful information to the administrator. The *Presenter* generates HTML documents for the Web-based user interface.

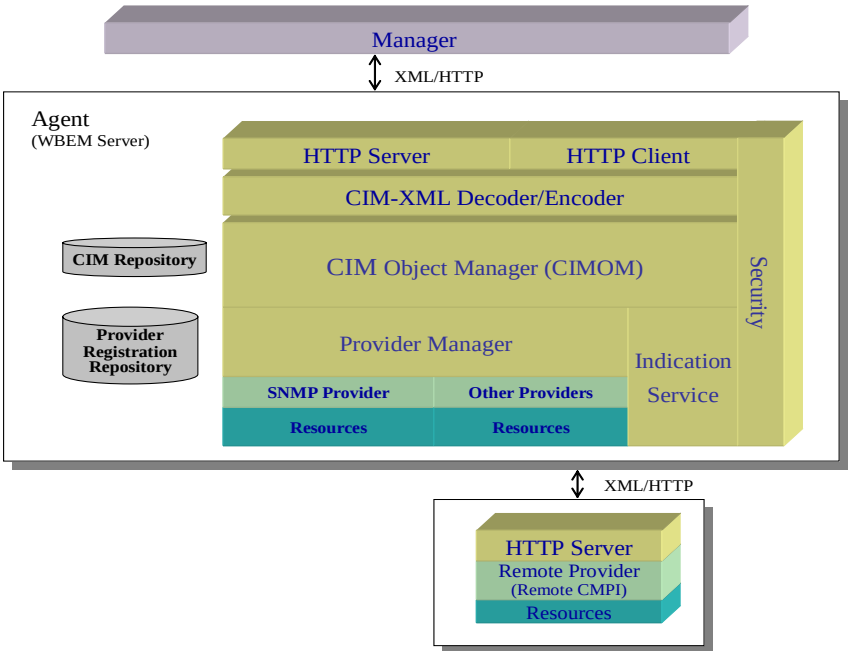

Figure 6. WBEM Agent Architecture

Figure 6 illustrates the agent architecture. The agent has both an *HTTP Server* and an *HTTP Client*. An *HTTP Server* is required for a manager to connect to the agent, and an *HTTP Client* is required for the agent to send notifications or appropriate messages to the manager. The agent converts a message that is sent from a WBEM client with the *CIM-XML Decoder*, and then delivers it to the *CIMOM*. The *CIMOM* attempts to retrieve the data requested by an administrator from the repository. If the data is not there, it gets the data from the provider that maps to the CIM operation that needs to be executed.

The *Provider Manager* is a component that controls various kinds of providers. The *Provider Manager* administers multiple providers that support different programming languages. These providers can be managed by being stored in the *Provider Registration Repository*.

The *Provider Manager* administers an *Instance Provider*, a *Method Provider*, an *Association Provider* and an *Indication Provider*. The *Instance Provider* creates/modifies/deletes an instance or gets/sets a property value. The *Method Provider* plays a role in invoking the methods such as the kill operation. The *Association Provider* can access an association or a reference class. The *Indication Provider* translates the occurrence of an event into a CIM indication and sends the indication to the *CIMOM* for further processing and delivery.

The *Provider Manager* also controls the SNMP Provider to integrate the existing SNMP devices. An *SNMP Provider* acts as a WBEM/SNMP gateway. It has to provide a specification translation through MIB2MOF or MIB2XML, and an interaction translation that can convert the WBEM's CIM operations to the SNMP's get, set and trap operations. The *Indication Service* processes the creation, modification and deletion operations for the indication subscriber data, which includes the list of clients who subscribe to

an indication.

The *CIMOM* manages both local providers and remote ones. The providers are easily deployed in a remote environment using the remote Common Manageability Programming Interface (CMPI) [11]. Remote CMPI enables providers to be run on remote systems without the need for an extra CIMOM. It uses a special proxy provider to relay requests to a remote location using so-called communication layers. The remote side has to start a daemon process which accepts remote requests and passes them on to the providers. The communication between a remote provider and the CIMOM is only for the transfer of the final result.

We have considered and incorporated the security issue in three areas: secure communication, user authentication and user authorization. First, we use HTTP over SSL (HTTPS) for the protocol between an agent and a manager. Second, basic authentication is provided by following the CIM-XML specification that supports its usage, configuring the client to pass the user ID and password to verify the user. Third, namespace authorization can be provided. If namespace authorization is enabled, a user must be granted the appropriate permission (i.e., read and write) for a target namespace.

5. WBEM Implementations

We plan to develop a WBEM manager and agent based on our design. Prior to our implementation, we have surveyed currently available WBEM products and implementations. First, we briefly describe the WBEM products.

Generally, WBEM is provided in the form that each vendor's network management system solution and application supports. In the case of IBM, WBEM is provided by its representative network and systems management solution suite called Tivoli [12]. It includes "Tivoli NetView", "Tivoli Enterprise", "Tivoli Manager for IBM Nways" and "Tivoli Cross-Sight". Cisco utilizes CIM in the enterprise network management tool called CiscoWorks2000 [13]. Sun provides Solaris WBEM Service and Sun WBEM SDK[14] for the Solaris Operating Environment. HP provides the WBEM solutions that include WBEM Services, WBEM Providers, HP WBEM Client and HP WBEM SDK [15].

Next, we introduce and compare open source WBEM implementations. We chose to examine Pegasus [16], WBEM Services [17], WMI, OpenWBEM [18] and SNIA CIMOM [19] since they are the representative WBEM implementations. Table 1 briefly compares the characteristics of the implementations such as licensing, developer, programming language used, OS platform, portability, documentation status and so on. All WBEM implementations presented in the table are open source except WMI. Currently, Pegasus and WBEM Services have the most active and biggest developer groups. As described in Table 1, WBEM Services has better portability than Pegasus because WBEM Services is JAVA-based. On the other hand, Pegasus provides more documentation than WBEM Services. Pegasus, OpenWBEM and WBEM Services implement most CIM operations described in the CIM specification. We explain each WBEM implementations more specifically in the next subsection.

	Pegasus	WBEM Services	WMI	OpenWBEM	SNIA CIMOM
License	MIT open source license	SISSL v1.2	Microsoft	BSD Style License	The SNIA Public License
Developer	The Open Group	Sun Microsystems Inc	Microsoft & EMC	Caldera International Inc	SNIA
Language	C++	Java	No records	C++	Java
Platform	Linux, AIX, HPUX, Windows Series	Any	Windows 98, NT, 2000, XP	Linux, Caldera Open Sun Solaris. FreeBSD and Mac OS	Any
Portability	Good	Excellent	Poor	Good	Excellent
Documentation	API, Developer's Document	API, Developer's Guide	No records	API	Developer's Document
Operation Implementation	Very good	Very good	No records	Very good	Poor
Activity	A lot of	A lot of	No records	Poor	Poor
Open Source	O	O	X	O	O
Provider	SBLIM	Inside API	Microsoft Standard Provider	SBLIM	Inside API or SBLIM

Table 1. The Comparison of WBEM Implementations

5.1 Pegasus

Pegasus is open source and implemented with reference to the technology of the WBEM architecture such as the CIM and CIM-XML standard. The Open Group (TOP) has been developing Pegasus, and it is actively participating in the WBEM standardization. Pegasus set its goal to providing many functions and being used by many servers. Many vendors are especially interested in Pegasus over other WBEM implementations because it possesses three desirable characteristics. Its first characteristic is that it has good portability because it is implemented to be suitable for multiple platforms and multiple programming languages. Second, it is a lightweight and efficient implementation because of its great regard for execution performance. Third, Pegasus' core functions are all modularized.

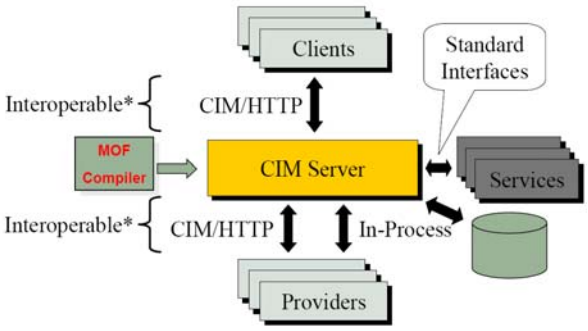

Figure 7. Pegasus Architecture

Pegasus's CIMOM supports the capacity to create, modify and delete a class, object, property, qualifier, etc. It also provides the functions that register each provider and deliver the client's request to

the provider along with a check for syntactic and semantic error. Each provider in Pegasus is defined in a class, which has its own method that is equivalent to the CIM operation. A provider manager converts the message delivered to the provider to a form that the provider can use. In addition, multiple language providers can be registered to the provider manager such as a C++ provider, a Java provider and so on.

5.2 WBEM Services

WBEM Services is driven by Sun Microsystems and implemented today in Solaris. WBEM Services has JAVA-based CIMOM and offers the JAVA API for a client and provider. When a client tries to get or modify information of a management object through the client API, the CIMOM responds to the client's request by interacting with the CIM repository or the related provider. The CIMOM also performs syntactic and semantic verification. The syntactic verification is to discover errors such as the omission of a semicolon or a brace, and the semantic verification is to find any logical errors in the program. The client application can send the objects to the CIMOM through the client API, and can request WBEM operations such as the creation and deletion of classes and instances in the CIM namespace.

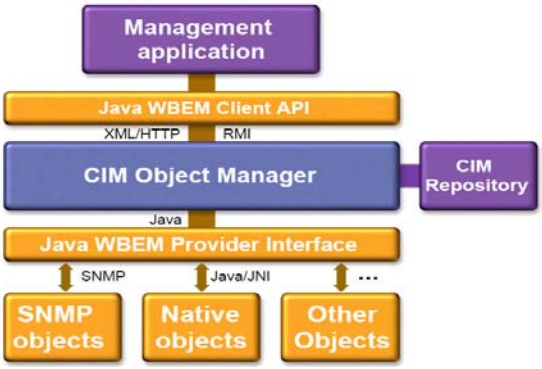

Figure 8. WBEM Services Architecture

The WBEM Services' providers exist as MOF files. These providers represent the systems, processes and resources such as CPU cycles, memory and so on. After the MOF files of each provider are compiled to a class by the MOF complier which is supported by the WBEM Services, each provider can then be available to the client. WBEM Services also provide a security service such as authentication and authorization. For users to be authenticated, the client users need to provide the user ID and password for the WBEM server. Also the WBEM server has an Access Control List (ACL) which has a list of users that allow to access to the WBEM server.

5.3 WMI

Windows Management Instrumentation (WMI) [20] is an implementation that has been developed by Microsoft. WMI also represents data in a consistent form and encapsulates it in an object-oriented way in the CIMOM repository. The CIMOM provides the convenience to collect and manipulate the information

of the management object. After a WMI provider collects the information from the managed objects, a management application can access those data through the CIMOM using the WMI API.

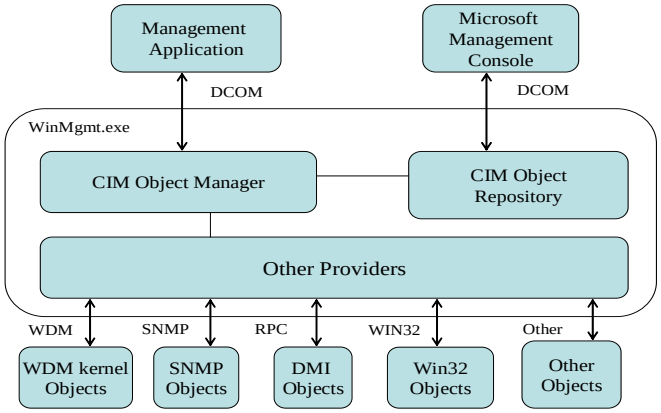

Figure 9. WMI Architecture

The WMI provider supports the Win32 environment, and it plays the role of intermediary between the CIMOM and the managed objects. A WMI provider gets the instrumentation data and converts it to the CIM schema. The Microsoft standard provider covers a performance monitor provider, a registry provider, a registry event provider, an SNMP provider, a Windows NT event log provider, a Win32 provider and a WDM provider. WMI verifies the user login information of local and remote computer using Windows 2000's security function. Access to CIM instances are only permitted to verified users. WMI establishes the security of system resources at the namespace level.

5.4 OpenWBEM and SNIA CIMOM

OpenWBEM is an open source that is driven by Center7 and is C++-based. OpenWBEM supports most CIM operations. Also, OpenWBEM has the advantage that it has a smaller repository than Pegasus. However, OpenWBEM implementation appears to have less industrial interest at this point. OpenWBEM's Standard Based Linux Instrumentation Manageability (SBLIM) [21] can be used in Pegasus, OpenWBEM and SNIA CIMOM.

SNIA CIMOM is driven by SNIA (Storage Networking Industry Association). SNIA CIMOM is a JAVA daemon and has no repository. SNIA CIMOM has had almost no research activity after 2002.

5.5 Interoperability among WBEM Implementations

Figure 10 illustrates the interoperability among Pegasus, WBEM Services and WMI. Pegasus CIMOM can be connected to a WBEM Services provider because Pegasus provider manager can manage providers that support different languages. Also, Pegasus CIMOM and WMI CIMOM are able to communicate. In addition, a Pegasus provider is able to correspond with a WBEM Services CIMOM or WMI CIMOM.

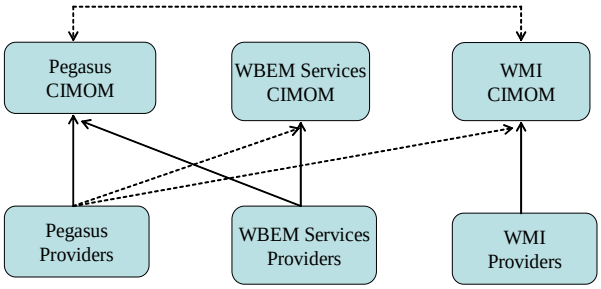

Figure 10. The Compatibility between WBEM Implementations

5.6 Benchmarking Test of WBEM Implementations

In this section, we selected Pegasus and WBEM Services among the open-source WBEM implementations and performed benchmarking tests. Pegasus and WBEM Services are the open sources in which developers are participating most actively, and they include WBEM standard management functions. Therefore, we considered these two open sources for implementing our U-server management system, and we performed the benchmarking tests to compare the performance between the two.

The test machine on which we installed Pegasus and WBEM Services was a Linux server with a Pentium IV 1.6GHz CPU and 256 MB RAM. Both the client and the server existed in the same machine and they communicated via HTTP. In the experiment below, we measured the latency of a request for each CIM operation 10 times, and averaged the results.

For the first experiment, we increased the number of classes to 100, 200, 400, 800, and 1600 and performed the enumerateInstances and enumerateInstanceNames operations. Figure 11 illustrates the Pegasus and WBEM Services latency. At that time, the number of instances that each class possessed was 100. We limited the number of each property and key to one.

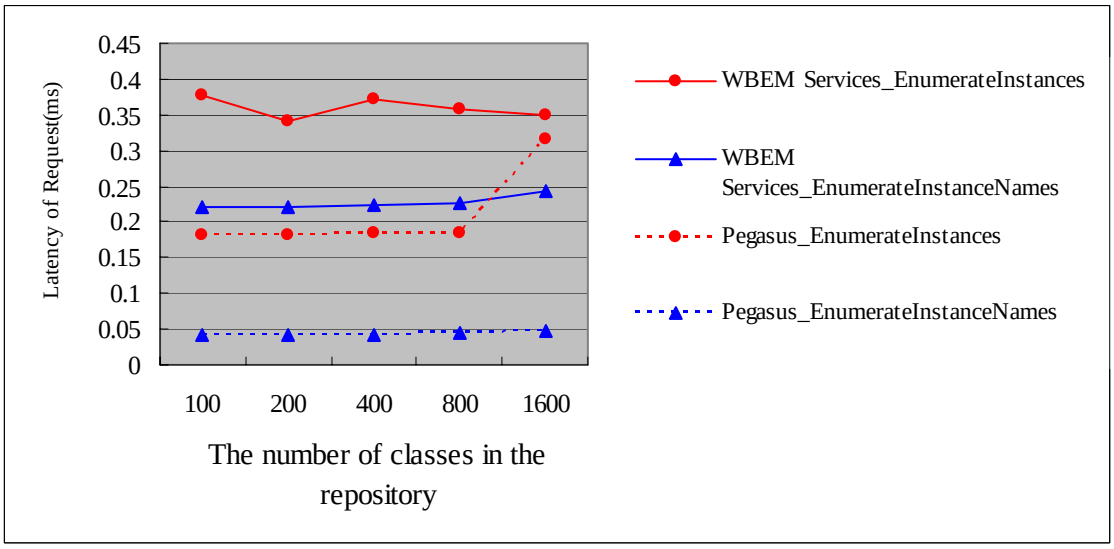

Figure 11. Result of First Benchmarking Test

As shown in Figure 11, we found Pegasus' latency was smaller than the WBEM Services' latency of request for the same operation; enumerateInstances and enumerateInstancesNames. Pegasus's performance was also better than WBEM Services' performance for other operations such as createInstance, etc.

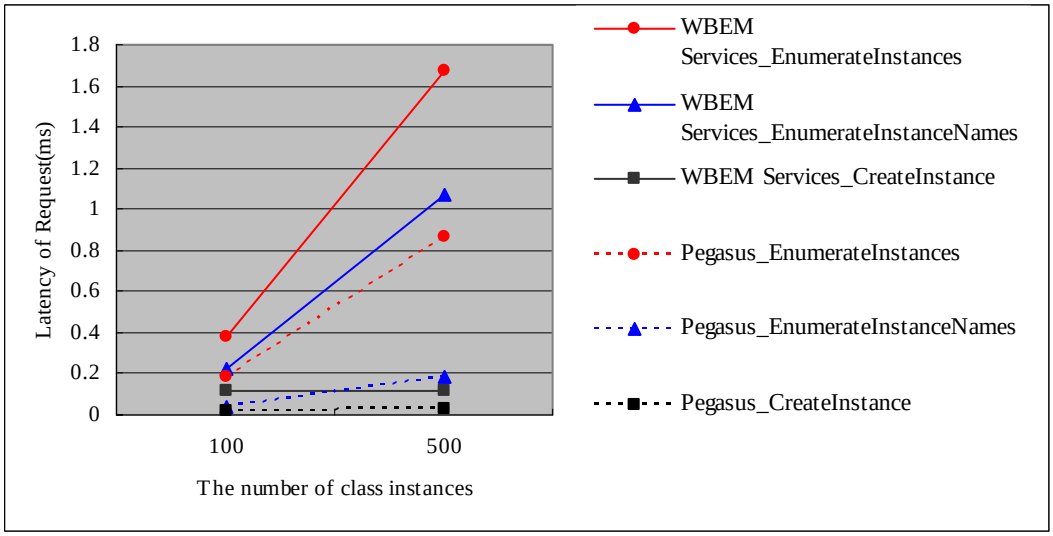

Figure 12. Result of Second Benchmarking Test

In the second experiment, we increased the number of instances from 100 to 500 in intervals of 100, holding the number of classes at 1600, and we measured the latency of a request. As described in Figure 12, the WBEM Services' latency increased suddenly according to the number of instances, whereas the Pegasus' latency of request increases slowly. These results show that Pegasus, which is implemented in C++, is better than WBEM Services, which is implemented in JAVA from the viewpoint of performance. We also changed the number of property or key values but the results did not change significantly.

Through the benchmarking tests, we found that Pegasus performs better than WBEM Services when processing CIM operations. In addition, Pegasus provides the CMPI API that supports a remote provider for a network device with little computing resources. Therefore, we are considering extending Pegasus' architecture for implementing our WBEM-based U-server management system.

6. Concluding Remarks

In this paper, we analyzed the requirements for managing ubiquitous computing servers and presented a design of a management system that satisfies those requirements using the WBEM technologies which are being standardized by DMTF. We described the management information using the CIM schema. We also used a CIM-XML encoding mechanism and CIM over HTTP for a transport protocol. We presented the overall architecture as well as the manager and agent architectures. For implementing our WBEM manager and agent, we considered the WBEM open sources and selected two

of them, Pegasus and WBEM Services. We performed benchmarking tests to compare the performance between the two, and found Pegasus to perform better than WBEM Services.

Therefore, our immediate future work is to implement the U-server management system that we have presented in this paper by extending the Pegasus open source. We will then work on a series of performance tests of Pegasus in order to evaluate and enhance our system. Through this testing, we will be able to validate the architecture and the WBEM implementation we have chosen.

References

[1] DMTF working group, http://www.dmtf.org.

[2] WBEM, “WBEM Initiative,” http://www.dmtf.org/wbem.

[3] DMTF, “Common Information Model (CIM),” http://www.dmtf.org/standards/standard_cim.php.

[4] Tim Bray, Jean Paoli and C. M. Sperberg-McQueen, “Extensible Markup Language (XML) 1.0,” W3 Recommendation REC-xml-19980210, Feb. 1998.

[5] DMTF, “Management Object Format (MOF),” http://www.dmtf.org/education/mof.

[6] DMTF, “Specification for the Representation of CIM in XML Version 2.0,” DMTF Specification, July 1999.

[7] DMTF, “Specification for CIM operations over HTTP Version 1.0,” DMTF Specification, Aug. 1999.

[8] Apache-SSL, http://www.apache-ssl.org.

[9] W3C, “XSL Transformations Version 1.0,” W3C Recommendation, Nov. 1999.

[10] W3C, “Document Object Model (DOM) Level 1 Specification,” W3C Recommendation, Apr. 2004.

[11] The Open Group, Pegasus Common Manageability Programming Interface (CMPI), http://www.openpegasus.org/uploads/40/4031/CMPI_Specification_13.pdf.

[12] IBM, Tivoli, http://www-306.ibm.com/software/tivoli/.

[13] Cisco, CiscoWorks2000, http://www.openview.hp.com/sso/isv/detail?appid=A516.

[14] Sun Microsystems, Solaris WBEM Service, http://wwws.sun.com/software/solaris/8/ds/ds- wbem24/.

[15] HP, HP WBEM Solutions, http://h71028.www7.hp.com/enterprise/cache/9920-0-0-225-121.aspx .

[16] Pegasus, http://www.pegasus.org.

[17] Sun Microsystems, Java Web Based Enterprise Management (WBEM Services), http://wbemservices.sourceforge.net.

[18] OpenWBEM, http://openwbem.org.

[19] SNIA CIMOM, http://opengroup.org/snia-cimom .

[20] Microsoft, Windows Management Instrumentation (WMI), http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/wmi_reference.asp

[21] IBM, Standard Based Linux Instrumentation Manageability (SBLIM) project, http://www124.ibm.com/sblim/

[22] J. H. Yoon, H. T. Ju and J. W. Hong, “Development of SNMP-XML Translator and Gateway for

XML-based Integrated Network Management," International Journal of Network Management, Vol. 13, No. 4, Jul.-Aug. 2003, pp. 259-276.

[23] Y. J. Oh, H.T. Ju, M.J. Choi, J.W. Hong, "Interaction Translation Methods for XML/SNMP Gateway," Proc. of DSOM 2002, Montreal Canada, Oct. 2002, pp.54-65.

[24] M. J. Choi, J. W. Hong and H. T. Ju, "XML-based Network Management for IP Networks," ETRI Journal, Vol. 25, No. 6, Dec. 2003. pp. 445-263.