A JMX Based Efficient Mobile Agents Platform Architecture for Network Management

Jung Hwan Lee[1] and Choong Seon Hong[2]

[1] School of Electronics and Information, Kyung Hee University, 449–701 Korea
nowsys@networking.kyunghee.ac.kr

[2] School of Electronics and Information, Kyung Hee University, 449–701 Korea
cshong@khu.ac.kr

Abstract. In order to overcome the disadvantages of existing centralized network management systems that use simple objects, dynamic object platform is proposed as alternative system. So the distributed network management systems are implemented using various distributed platforms such as CORBA and JAVA-RMI. Subsequently mobile agent-based platform is proposed. The mobile agent-based platform can additionally provide flexibility and scalability to network management system which CORBA or JAVA-RMI based platform do not support. In this paper, we address the architecture to solve the problem of the occurrence of additional traffic by using mobile agents and to save resources of network element. This paper makes a description about efficient network management architecture using mobile agents. Also we design agents using information architecture of TMN for efficient resource management of network element and improvement of operation performance.

1 Introduction

With popularization of Internet, network traffic is increasing continuously. Network can be highly jammed and can be caused to the delay of response time by effect of such increasing network traffic. It has become an essential work now that to manage network continuously and monitor, analyze and solve these problems. A lot of systems for the network management have been proposed by this necessity. ISO (International Organization for Standardization) defines CMIS/CMIP (Common Management Information Services / Common Management Information Protocol) [1]. And IETF (Internet Engineering Task Force) defines SNMP (Simple Network Management Protocol) [2]. These services and protocols are typical centralized structure using a client / server model. SNMP is used in various network management systems because the architecture that is divided as manager and agent is relatively simple and is a clear architecture. Also SNMP implementation is not difficult. So many network de

I. Chong (Ed.): ICOIN 2002, LNCS 2344, pp. 768–779, 2002.

vice vendors implement these modules and the modules are loaded into network device. However, in a network such as WAN (Wide Area Network), it has a difficulty to monitor several sub-network because of bottleneck [3]. The system with decentralized architecture (Distributed Architecture) that supports MAS (Multi-Agent Systems) [4] was proposed to solve the problem of centralized management architecture.
It is a static object platform which several distributed objects are fixed in each agent using CORBA or Java-RMI. And in a contrast to this, we call it a dynamic object platform that using mobile agents. The advantages of dynamic object platform compared to static object platform are flexibility and scalability. In this paper, we make use of advantages of mobile agents which include dynamic object platform. First we studied about the weak point that mobile agents can have, namely additional traffic occurrence problem that would happen when mobile agents move. Then we studied about efficient resource management work that proposed for network management using mobile agents in chapter 2. In chapter 3, we describe about mobile agents design and stationary agents design. And in chapter 4, we describe about design and implementation of new platform architecture based on proposed item. At last, we explain about conclusion and forward subject.

2 Network Management Platform Using Mobile Agents

In this section we will describe the existing research that is proposed for network management using mobile agents platform. If we use mobile agents in network management, we can reduce unnecessary traffic over network to use network management classification factors that fault management, account management, configuration management, performance management and security management can reduce the use of unnecessary resources network element. Also if we use a java platform (JVM), it supports heterogeneous environment in network and network device. Table 1 shows merits and demerits that can have about network management using mobile agents.

Table 1. Merits and demerits of network management system using mobile agents

Item	Contents
Advantages	Efficient saving
	Support for heterogeneous environment
	Storage saving of network element by mobile agents transfer
	Extensibility
	Easy software upgrade
Disadvantages	Additional traffic occurrence when mobile agents move
	Transfer domain specification of mobile agents for mobility guarantee

2.1 Platform Architecture Applied in Network Management

General architecture is shown as figure 1. Mobile agents that are created by the network manager migrate to target node that needs the management.

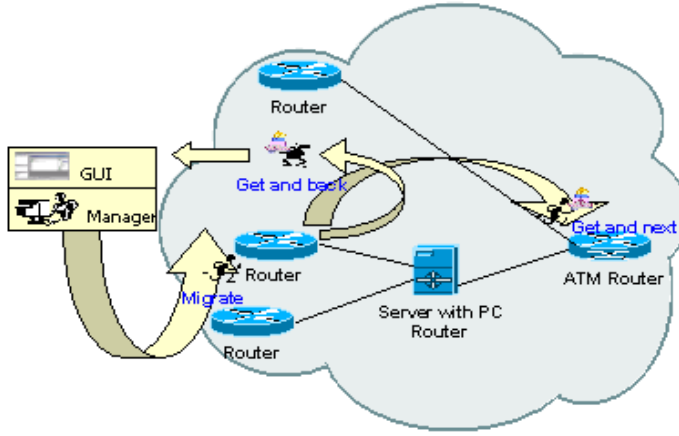

Fig. 1. General architecture that use mobile agents in network management

The general architecture which is consisted of simple operations such as GnG (Get and Go, Get and Back) [3] has a difficulty in management of mobile agents, performance management of network and efficient management of data. So the platform having new algorithms and structure proposed in order to supplement these problems.

2.2 MIAMI Project

The objectives of MIAMI (Mobile Intelligent Agents in the Management of the Information Infrastructure) [5] project are to examine the applicability of Mobile Intelligent Agents (MIAs) to network and service management. The MIAMI project, within this context, focuses on the following key objectives: [6]

- Create a unified mobile intelligent (MIA) framework by validating, refining and enhancing the OMG MASIF standards according to the requirements for an Open European Information Infrastructure.
- Develop mobile intelligent agent (MIA) based solutions for the management of the Open EII and for the provision of advanced communication and information services.
- Create a reference implementation of the Unified MIA Framework and solutions in order to evaluate the service solutions in a Pan-European business environment.
- Produce the following recommendations :
  - To infrastructure and terminal providers : when, where and how to introduce the MASIF in their future products

– To service provider : how to develop MIA based solutions for the management of the EII and for the provision of advanced communication and information service.
– Augment the results of selected ACTS projects by the integration of MIA based solution.
– Participation in the “Domain 5” cluster of ACTS agent projects for the coordination of standardization activities.
– Disseminate results by demonstrations, publications, providing input to standardization bodies, industrial forums.

In this research, the software agents which have a constrained mobility placed to solve problems of legacy mobile agents platform and to manage more effectively about network. And they proposed dynamic and efficient mobile agents platform (figure 2) that allows network’s dynamic resource management and functional delimit of each management element through AVP (Active Virtual Pipe).

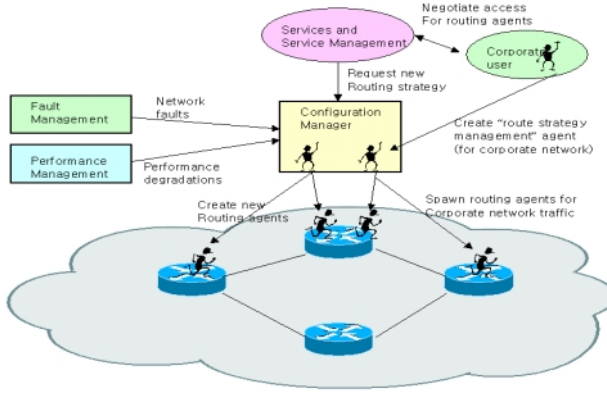

Fig. 2. The architecture of MIAMI platform

Also figure 3 shows an AVP of MIAMI. DCM (Dynamic Connectivity Manager) provides dynamic connection to each management factors (i.e., configuration management, performance management, fault management) according to types of mobile agents.

3 Design and Implementation of Agents

The agents is divided into two classifications in mobile agents platform. The one is a mobile agent and the other is a stationary agent. The mobile agents have a mobility and they can visit destination node or target node and collect necessary information. While stationary agents can’t move to other area or node but it can manage dynami-

cally specific area with mobile agents. And we basically use information architecture of TMN basically for all efficient management of agents.

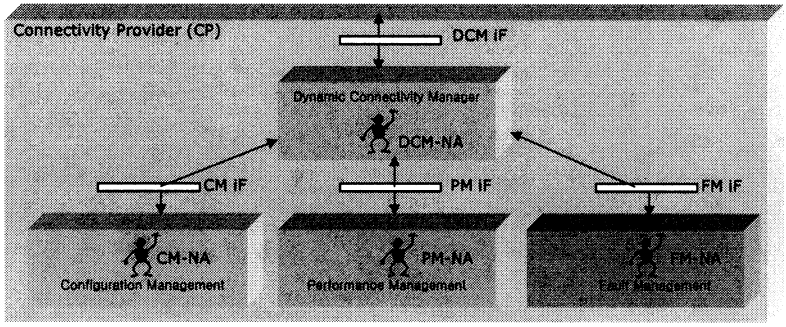

Fig. 3. MIAMI Active Virtual Pipe Domain

Also we divided by each fault management, configuration management, performance management, account management and security management using Management Information model of OSI [7]. We apply the information architecture of TMN [8] to all of agents for efficient management. Figure 4 shows Information Architecture of TMN. And we define the stationary agents class of each management using the functional management area of OSI

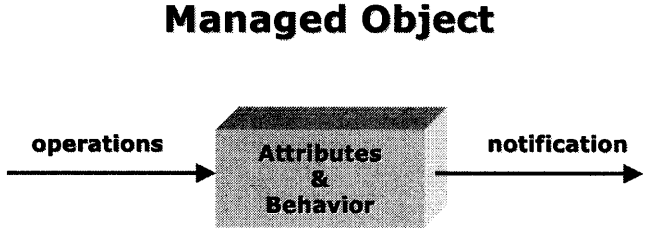

Fig. 4. Information Architecture of TMN

3.1 Design of Mobile Agents

As a mobile agents platform, we considered IKV++ company's Grasshopper platform which was strongly recommended in a result report of MIAMI project. Although there are so many platforms for mobile agent which was provided by different vendors such as Aglets [9], Grasshopper [10], Voyager [11]. But Grasshopper is a mobile agents platform that is built standard of the Object Management Group (OMG). (i.e., Mobile Agents System Interoperability Facility (MASIF)). The MASIF standard has been initiated in order to achieve interoperability between mobile agents platforms of different manufactures [12]. Grasshopper platform is composed of regions, places, agencies and different types of agents

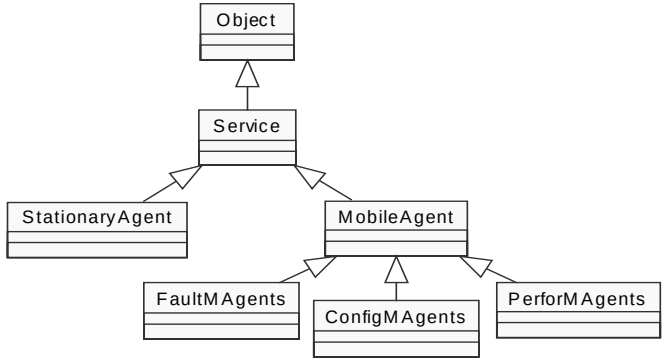

Fig. 5. Class hierarchy diagram of mobile agents.

Basically, mobile agents is object which has the operations and attributes. Therefore, mobile agents can reduce unnecessary operations and attributes so that reduce quantity of traffic which may happen additionally. So, we divided to 5 classification items that refer mobile agents over. And we subdivided by 3 classification (figure 5) that mobile agents can be applied usefully among them. Such classed mobile agents communicates with stationary agents that has associated data. (table 2) And mobile agents is created by classification according to manager's request and is sent to network element.

3.2 Design of Stationary Agents

We implemented the stationary agents using JMX (Java Management Extension) [13]. The Java Management extensions define architecture, the design patterns, the APIs, and the services for application and network management in the Java programming language. The JMX architecture is divided into three levels. (figure 6)

- Instrumentation level
- Agent level
- Distributed services level

The instrumentation level provides a specification for implementing JMX manageable resources [14]. A JMX manageable resource can be an application, an implementation of a service, a device, a user, and so forth. The instrumentation of a given resource is provided by one or more Managed Beans, or MBeans, which are either standard or dynamic. Standard MBeans are Java objects that conform to certain design patterns derived from the JavaBeans component model. Basic structure of JMX is as following. MBeans model has a similarities with Information Architecture of TMN. Information Architecture TMN uses an object oriented approach and is based on Management Information Model of OSI.

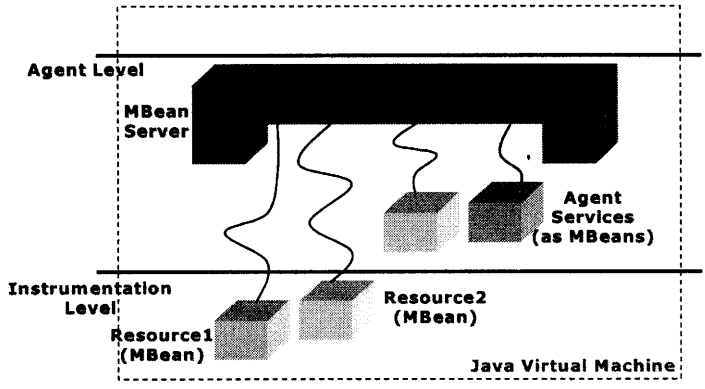

Fig. 6. Basic architecture of JMX

We designed MBeans model for efficient management of stationary agents. And we defined three MBeans. Defined MBeans are PerforMSAgents, ConfigMSAgents, FaultMSAgents. Figure 7 shows the class hierarchy diagram of stationary agents.

MIB values are defined in RFC 1213 [15]. We refractionate MIB which defined in RFC 1213 for management of MBeans by OSI functional area (FCAPS). Classifications are same as next table 2 [16]. Classified values are managed by each MBeans. And each MBeans have an operations and attributes. Also ManagedStationaryAgents contacts with PerforMSAgents, ConfigMSAgents, FaultMSAgents. SNMP API also is supported in JMX. JMX smart agents are capable of being managed through HTML browsers or by various management protocols such as SNMP and WBEM [17].

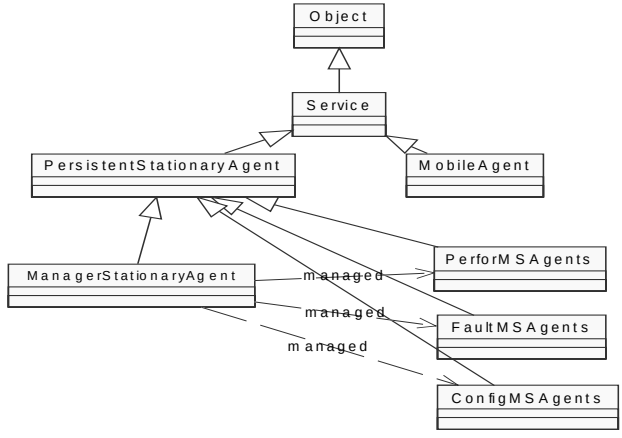

Fig. 7. Class hierarchy diagram of stationary agents

Stationary agents above in class hierarchy take the role of MBeans in JMX. Also, all stationary agents can do resources management of network elements using a persistent service of Grasshopper.

Table 2. The rearrangement of MIB

Agents Names	MIB Variables
PerforMSAgents	SnmpInPkts, snmpOutPkts, sysUpTime, ipDefaultTTL, ifInDiscards, ifOutDiscards, iFInErrors, ifOutErrors, ifInOctets, ifOutOctets, ifInUcastPkts, ifOutUcastPkts, ifInNUcastPkts, ifOutNUcastPkts, ifInUnknownProtos, ifOutQLen, ipInReceives, ipInHdrErrors, ipForwDatagrams, ipInUnknownProtos, ipInAddrErrors, ipInDiscardsm upInDelivers, ipOutDiscards, ipOutNoRoutes, ipRoutingDiscards, ipReasmReqds, ipReasmOKs, ipReasmFails, ipFragOKs, ipFragFails, ipFragCreates
FaultMSAgents	SysObjectID, sysServices, sysUptime, upInHdrErrors, ipInAddrErrors, upReasmFails, ipInReceives, ipForwDatagrams, ipInDelivers, upOutRequest, ipOutDiscards, ipOutNoRoutes, ipRoutingDiscards, ipReasmReqds, ipReasmOKs, ipReasmFails, ipFragOKs, ipFragFails, ipFragCreates
ConfigMSAgents	SysDescr, sysLocation, sysName, ifDescr, ifType, ifMtu, ifSpeed, ifAdminStatus, ipFowarding, ipAddrTable, ipRouteTable

4 Design and Implementation of Proposed Platform Architecture

As explain in chapter 3, whole platform is consisted of mobile agents and stationary agents. The whole platform is same as figure 8.

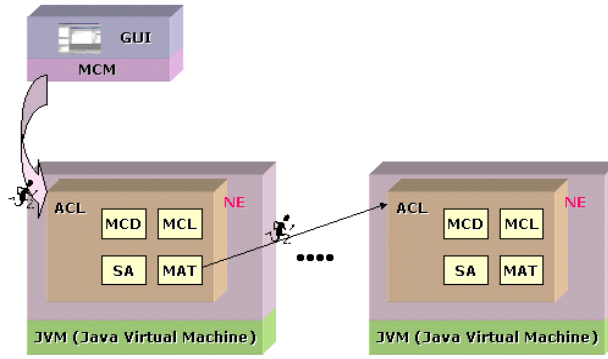

Fig. 8. Proposed platform component diagram

4.1 MCM (Mobile Code Manager)

Mobile code manager refers code repository and creates mobile agents by manager request. And Mobile code manager sends mobile agents to target node. The module composition of MCM is as following.

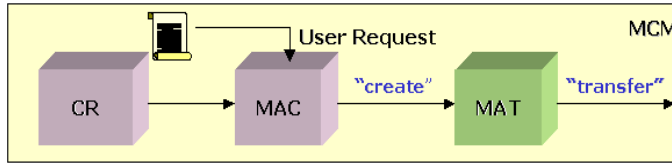

Fig. 9. MCM(Mobile code manager) component diagram

■ CR (Code Repository) : When mobile code manager creates mobile agents, code repository is referred by user request for functional network management. It stores operations and attributes for mobile agents.
■ MAC (Mobile Agents Creator) : Refer code repository and user code, it creates mobile agents.
■ MAT (Mobile Agents Transfer) : Transfer mobile agents to target node. (It depends on Grasshopper platform).

4.2 ACL (Active Class Loader)

This module exists in network elements, doing a practical network management job with mobile agents

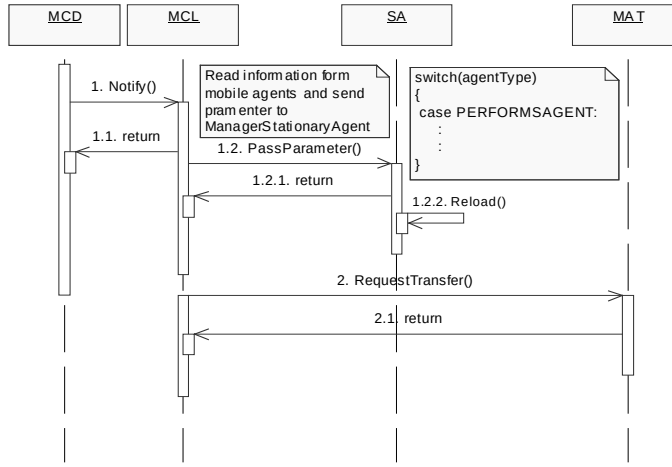

Fig. 10. Sequence diagram of Active Class Loader

■ MCD (Mobile Code Daemon) : If mobile agents arrive in destination node, alarm to node.
■ MCL (Mobile Class Loader) : Read information of mobile agents and pass data that need in stationary agents management to SA
■ MAT (Mobile Agents Transfer) : Move into other target node of mobile agents

■ SA (Stationary Agents) : Stationary agents manages practical components of network element. And stationary agents improves resources management of network element efficiency using persistent service of Grasshopper

5 Simulation Results

We compare the memory availability of stationary agents and code size of mobile agents for evaluation of proposed platform architecture. Simulation environments are next.

- OS : Windows 2000 Professional
- PC : Pentium III 733Mhz, RAM 512M
- Tool : JDK 1.2.2, JDMK 4.2, Grasshopper Platform 2.2.2,

It is an optimized code to perform network management operation for mobile agents as shown in figure 5. The mobile agents code is a java class file, so we can remove unnecessary operations and attributes of mobile agents. And it can make the code more minimum size and optimum size by applying classification of functional management area.

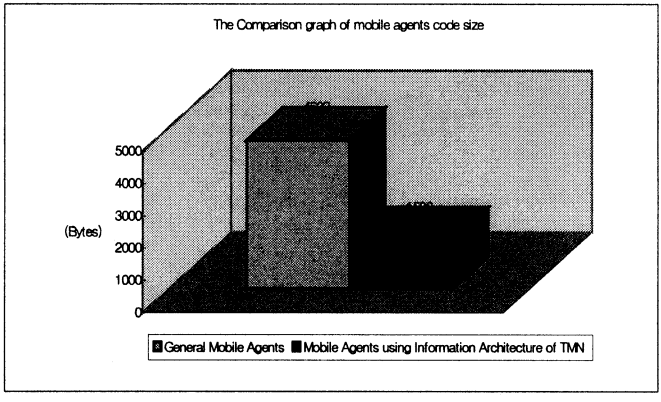

Fig. 11. Byte size comparison graph

In the above figure 11, the left bar means average code size which does not apply classification of functional management area. The right one is an average code size which applied classification of functional management area. The size of mobile agents which is applied classification of functional management area decrease about 53%. So it only needed a minimum time cost to move a destination node or manager.

Also we use a persistent service for efficient management of stationary agents. The persistence service is a part of the core functionality of Grasshopper agencies. Its purpose is to persistently store the data states of all currently hosted agents as well as

runtime information about all places that exists on the agency. Persistent service deallocates stationary agents that it is no present necessity in memory. And necessary stationary agents reallocate. Because main work of network element is packet routing, it is all-important that reduces additional resources assignment. We can see the memory measurement and number of MBeans in table 3. The API of the persistence service is divided into two parts, one provided by the agency (interface de.ikv. grasshopper.agency.IAgentSystem) and the other provided by the persistence-supporting agent (classes Persistent MobileAgent, PersistentStationayAgent).

Table 3. The comparison of memory availability
(#N : number of stationary agents, T.M : total memory, A.M : Available memory)

	# N	T. M	A. M
SA with persistent service	3	523,744KB	231,054KB
SA without persistent service	3	523,744KB	224,529KB

6 Conclusions

The management platform applying mobile agents are flexible and extendible more than other network management platforms. But the mobile agents platform can not be applied to most network management actually up to now. Because the JVM is so heavy to integrates with node OS and platform dependency of itself. Such problems can be solved through Java Chip and so on if consider that JVM is developed by embedded system [18]. Mobile agents platform which proposed in this paper uses OSI management function classification based on Information Architecture of TMN. Also, we studied a way to minimize code of mobile agents and minimize network traffic that may happen additionally. The new schemes which apply persistent service to MBeans of EJB Based and to managed MIB variables by each stationary agents are more efficient in resources management of network element. Our researches till now described a general platform that use mobile agents.

For our future works, we will research algorithms about transfer path decision problem of mobile agents. And we will study about the network management scopes using mobile agents.

Acknowledgements. This work was supported by grant No. 2001-1-30300-001-2 from the Basic Research Program of Korea Science & Engineering Foundation.

References

1. IETF, “The Common Management Information Services and Protocols for the Internet (CMOT and CMIP)”, RFC 1189, http://www.ietf.org
2. IETF, “A Simple Network Management Protocol”, RFC 1157, http://www.ietf.org
3. D. Gavalas, M Ghanbari, M. O’Mahony, D. Greenwood, “Enabling Mobile Agents Technology for Intelligent Bulk Management Data Filtering”, NOMS 2000, 623p, April 2000
4. A. Bieszczad, B. Pagurek, T. White, “Mobile Agents for Network Management”, IEEE Communications Survey, Vol 1, No. 1
5. MIAMI project, “Mobile Agent Platform Assessment Report”, http://www.fokus.gmd.de/research/cc/ecco/climate/documents/miami-agplatf.pdf
6. The MIAMI project at UCL, http://www.ee.ucl.uk/~dgriffin/miami/
7. Management Information Protocol of OSI, ISO DIS 10165-1: "Information Processing Systems – Open Systems Interconnection – Structure of Management Information – Part 1: Management Information Model", Geneva, 1993
8. Information Architecture of TMN, http://snmp.cs.utwente.nl/tutorials/tmn/index-15.html
9. IBM Aglets, http://www.trl.ibm.com/aglets/
10. IKV++ Grasshopper 2, http://www.grasshopper.de/index.html
11. ObjectSpace Voyager, http://www.objectspace.com/products/voyager/
12. MASIF, http://www.det.ua.pt/Projects/difference/work/D7/d7chap4.html
13. JMX, http://java.sun.com/products/JavaManagement/
14. EJB, http://java.sun.com/products/ejb/
15. IETF, “Management Information Base for Network Management of TCP/IP-based internets : MIB-II”, RFC 1213, http://www.ietf.org
16. Mi-Young Kang, Sung Kim, Cheul-Young Kim, Ji-Seung Nam, “The design and implmentation of functional class for real-time network management based on java”, Korea information science society, Vol. 28, No 1, April 2001
17. Web-based Enterprise Management, http://www.dmtf.org/standards/standard_wbem.php
18. Embedded Java, http://java.sun.com/products/embeddedjava/