IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITP4557

JMX-Based Network Service Management

Guilan Kong, Boyi Xu, and Lan Fang
Beijing Institute of System Engineering
Mailbox 9702-19, Beijing, 100101, China
Tel: 86-10-66356598 Fax: 86-10-64836117
glkong2002@yahoo.com.cn

ABSTRACT

Service Level Management is an important concept and implementation on network emerging as the times require. But the state-of-the-science SLM in nature is Network Service Management. JMX is a type of dynamic management technology. CIM is a set of standards-based resource and instrumentation models under the auspices of the DMTF. JMX and CIM have been widely used in the area of Network Service Management. This paper deals with our project of integrating JMX, CIM technologies into Network Service Management framework.

1. INTRODUCTION

1.1. Motivation for Network Service Management

As network is growing larger and the services it provides are growing more comprehensive, rapid and flexible providing of services, coupled with ensuring that these services are highly available is the most important guarantee to network users. It is absolutely vital to keep services highly available and accessible, as downtime in e-commerce will be translated into the loss of productivity, opportunities, revenues and profits. These requirements have been accelerating the evolution of Network Service Management.

1.2. Definition of Network Service Management

Traditional network management systems are focused on network elements such as switches and routers, but nowadays, network management, system management and application management trend toward being integrated together. Because service consumers today are mostly Internet surfers, service management is considered as a branch of network infrastructure management.

Network Service Management is focused on the management of the services that network provides, such as database access, DNS, DHCP, etc., although information on network elements is also available. In general, NSM includes data collection, status polling, event processing and status report.

1.3. Technical Background

To face new challenges to manage service-driven network environment, some new management technologies and standards come forth. There are several candidates for becoming the new management protocol or technology of choice: SNMPv3, CMIP, JMX, WBM, WBEM. As a type of open, standard, portable and scalable technology, Java Management Extension (JMX) has been regarded as the most suitable technology in the area of Network Service Management. Web-based Enterprise Management (WBEM) is an initiative by DMTF trying to integrate all these standards. WBEM consists of four components: Web client, Common Information Model Object Manager (CIMOM), CIM repository and CIM data provider. Combining JMX with CIM is a good choice to implement Network Service Management.

2. FRAMEWORK OF NETWORK SERVICE MANAGEMENT

The goal of Network Service Management is to monitor and maintain network services. The manageable resources of network services include the status and the environment of services. The status of one service comprises the accessibility, the response delay and the exact downtime of the service. The environment of one service comprises IP address of the host, running port and the manager of the service, etc. Network Service Management involves an activity of monitoring network services, auto-polling of network services and the data collection of services. For Internet users, the majority of services are provided on IP network, so NSM in our project is IP-centric.

In our project, we design the framework of NSM to have four multi-level components: the top level is master (management application); the middle-level is distributed manager (MBean server) and database; the bottom level is service agent (data collector) on the managed host. Master is for administrator to control and configure the whole management system, and it provides management information browsing to both administrator and general users. Distributed manager provides core services management functions, such as status polling, data collection, interaction of service agent and distributed manager, etc. In our design, we split core functions on distributed manager into five parts: discovery, capabilities checker, data collector, status polling and service reporter. Service agent aims at the data collection of the managed service. Database services for the storage of collected data.

In the following, we'll discuss more about the functions developed on the distributed manager. Firstly, discovery in NSM consists of two parts: one is to discover an IP address, the other is to discover the services supported by that IP address. In fact, what the discovery process does is to generate an event implying a new IP address has been discovered. Secondly, capabilities checker is responsible for discovering all the services that can be monitored, such as http, DNS, etc. Thirdly, there are two major ways that NSM collects data about the network services. The first is through polling. Polling process connects to one network service and performs a simple test to see if the service is responding correctly. If not, events are generated. The second is through service agent. The polling process will only operate on services that have been previously discovered by capabilities checker. Lastly, the final process is service reporter. By analyzing data stored in the database, this process can show users the current status of services through web browser.

The framework of NSM is shown as Figure 1.

We plan to use JMX architecture to implement the NSM framework discussed above. And as CIM is adopted widely in the area of system management, we plan to use CIM to model manageable services, and to develop service agents communicating with CIMOM to get or set data about managed services. Wrapped by Java, CIM-based service agent can be a component of JMX architecture.

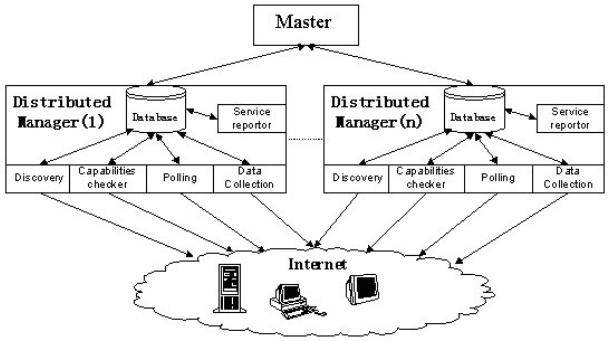

Figure 1 Network Service Management Framework

ment Beans (MBeans) are employed to instrument resources being managed. Simply put, an MBean provides a programmatic interface for managing one network resource. MBeans come in two flavors: Static and dynamic. Dynamic MBeans can change their behavior at run time.

3.2. Agent level

The JMX Agent level provides management agent. The main component of an agent is the MBean server. The MBean server is used to register and interact with MBeans. At this time, there is no direct access to an MBean, the MBean server hides their implementation behind a standardized management interface and mediates between all different interactions. The MBean server can be dynamically extended by adding MBeans. There is one MBean server per JVM. Also defined at the agent Level are protocol adaptors that enable external applications to interact with the MBean server through various protocols. Protocol adaptors must be implemented as MBeans in order to register with the MBean server.

3.3. Manager level

The JMX Manager level is the domain of custom management applications that can operate as a manager for distribution and consolidation of management services. Another JMX manager interfaces with agent via the connector, and support for different management application protocols is achieved at the agent Level using protocol adaptors. JMX manager interfaces with external database using JDBC.

JMX defines additional APIs in supplementary specifications for some common management protocols. Using these APIs, we can build capabilities at the JMX Manager level and Agent level to bridge to existing management system.

Figure 2 depicts the JMX management model.

In addition, JMX provides a number of Java APIs for existing standard management protocols. These APIs are independent of the three-level model, yet they are essential because they enable JMX applications in the Java programming language to link with existing management technologies.

The JMX architecture is as Figure 2.

4. INTEGRATING JMX INTO NSM

4.1. Necessity of JMX and CIM in NSM

The growth of the Internet has given rise to many new technologies to serve the increasing demands of the current generation of applications. JMX is emerging as the preferred way to build Network Service Management system. Chief among the reasons for this growing trend is the ease of this new technology. JMX provides a flexible, distributed, and dynamic management infrastructure to be implemented. The key features JMX provides are: Platform independence, Protocol independence, Information model independence and Management application independence.

In NSM framework described above, there are two crucial sides. One is the core functions developed on the distributed manager. They need a message or event mechanism to coordinate them and they must be developed to interact with manager easily (MBean server can manage MBeans well). The other is the data collection of managed services. As we all know, polling process is just for the accessibility and response delay of services, so CIM-based service agent is the key process for the data collection of the environment of one service.

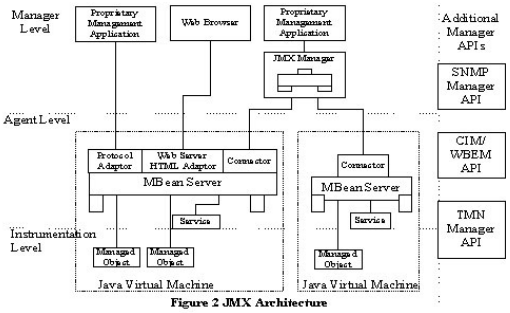

Figure 2 JMX Architecture

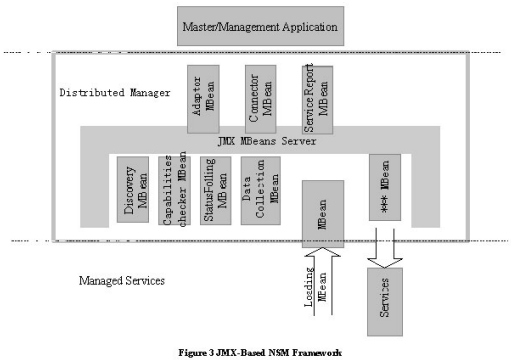

Figure 3 JMX-Based NSM Framework

Considering the advantage of JMX technology, we integrate JMX architecture into NSM framework.

4.2. Integrate JMX Architecture into NSM Framework

In JMX architecture, the instrumentation level instruments the manageable resources by standard or dynamic MBean, then register to the agent (MBean server), and agent plays the role to coordinate MBeans and lets management applications connect and interact with all MBeans, then communicates with the manager through connector or protocol adaptor.

By integrating JMX architecture into NSM, we instrument the core functions on distributed manager such as discovery, capabilities checker, status polling, data collector and service reporter through MBeans, and register these MBeans to an MBean server, and through the message mechanism of MBean server they can communicate with each other.

Figure 3 shows this framework.

4.3. Integrating CIM-based service agent with JMX

In our research, we call those applications that reside on service host to get or set data of managed services service agent. It's an important way to implement data collection through service agent. Since JMX provides SNMP API for developer in network management field, when a service host supports SNMP, we can develop a SNMP adaptor to communicate with SNMP agent on remote host to get or set MIB information. But what should we do if the service host does not support SNMP? In this instance, the principal problem is how to model manageable services and instrument the application that collects data of managed service. In our project, we use CIM standard to model managed services, and develop CIM data provider of each managed service, then we develop CIM-based application as a service agent to communicate with CIMOM to get or set data of each managed service. The next work is to use Java to wrap CIM-based application (if CIM data provider and CIMOM not implemented in Java, it will be a big problem, such as WMI). JMX provides WBEM API for CIM/WBEM developers. We can use this API to develop CIM adaptor MBean on the distributed manager to communicate with the CIM-based service agent. Currently, we have implemented data collector on distributed manager for SNMP service.

Figure 4 shows the structure of JMX and CIM working together.

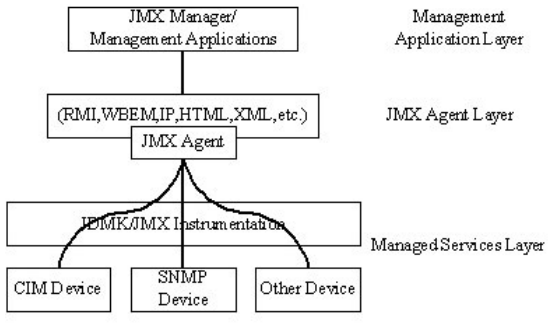

Figure 4 JMX and CIM working together

5. CONCLUSION

Network Service Management is the fundamental implementation of the SLM. JMX, as a 3rd generation management technology, brings deployment flexibility through protocol independence and dynamic extensibility and scalability.

CIM is a most comprehensive Information Model to define services data. To use CIM to instrument network, system, application, service, etc. is the future trend of network management.

Combining JMX with CIM can trigger a great advancement in network management. Development of telecommunication network services management is in progress.

6. REFERENCES

1. Java Dynamic Management Kit (JDMK) 2.0 White Paper, *Dynamic Management for the Service Age.*
2. http://www.openwings.org/openwings-0.9/tutorial, Openwings Management Service Specification Alpha Ver 0.72.
3. *Java Management Extensions (JMX)*, Christophe Ebro
4. *JMX Specification Lead*, Sun Microsystems, Inc.
5. http://www.opennms.net/users/docs/docs/html/part2.html
6. http://www.opennms.net/users/docs/docs/html/part3.html
7. http://www.opennms.net/users/docs/docs/html/part4.html
8. http://ca.com/products/descriptions/tng_slm.pdf
9. http://www.iseeman.com/docs/wmi.html