Automatic Injection of Management Mechanism to Applications

PING-JER YEH, RUEY-SHYANG WU, SHYAN-MING YUAN
Department of Computer Science
National Chiao Tung University
1001, Ta Hsueh Road, Hsinchu 300
TAIWAN

Abstract: To have flexible and standard application management infrastructures, many industry specifications are defined. Most of them define a lot of interfaces and functions to be implemented by the managed applications, but they also increase the development effort, especially when applications are more and more complex today. We need an approach to adding management functions more efficiently. In the paper, we propose an approach to adding management mechanism to applications automatically. System developers will not care about the management functions. If the application encounters any problems, the management function will detect the error and pass related information to the management system automatically. Then, the management system will know where the problem is. Adding the management function does not have to modify the program code. It only needs to analyze managed resources and find the underlying APIs. Through the method, there will be no extra effort for system design and develop. Distributed system management will become easy.

Key-Words: Application and resource management, hook, probes

1 Introduction
Distributed system management is harder and more important than traditional system management. It is also more complex because there are many heterogeneous systems. It is more important because any single fault will affect the whole system. Therefore, distributed system management should have more flexible architecture and standard protocol; the former can adopt heterogeneous systems into the consistent architecture and the latter can let different systems exchange information. Several industry standards can cover both of them, such as SNMP, CIM, and JMX. Those standards are widely used by hardware and software vendors. These vendors produce manageable hardware and software. They also develop management consoles to control these hardware and software.

Those management standards define several interfaces those hardware and software should follow. Without implementing those interfaces, those devices and applications will not be managed. It will increase developing effort especially those interfaces are not simple. Designer and developer should consider embedding management functions into their systems and developing them very carefully. Otherwise, errors may occur in management functions rather but not original system functions. The situation is more serious for distributed systems. Distributed systems have to follow existing architecture and framework to solve distributed problems. Those problems include system communications, fault tolerance, transaction, etc. For this reason, it will reduce developing effort if the management functions can be discarded.

In this paper, we propose a solution to add management mechanism into applications without adding or modifying existing program codes. We analyze the management resources and application behavior. Resources include CPU, memory, network, etc. Application behavior is the action performed to solve problems. Based on the analysis, two different monitor methods are proposed. It will detect errors and get the related information. If there is abnormal situation, the error notification mechanisms will startup. The action can be to notify system administrator, to isolate the fault application or to recover from error by pre-defined rules. The other applications will not be affected by the failure and problems can be identified quickly. Using the architecture will make it easy to create manageable distributed environment.

The rest of this paper is organized as follows. Chapter 2 is the background, related works and objectives. Chapter 3 introduces the analysis of manageable resources and application behavior. Chapter 4 illustrates system implementation and Chapter 5 is the conclusions and future works.

2 Background and Objectives
There are many industry standards are widely used from hardware and software vendors. They provide well-designed protocol and architecture.
2.1 Related Standards
Currently, there are many management standards and applications. Most of them are client-server, agent-manager, or provider-consumer based. Users can use the management console to monitor and control the whole environments. They all need to install or implement a specified interfaces and protocols on the client or provider. We call the monitor application as management agent. How to implement the management agent is the key to realize system management.

The most popular management protocol is Simple Network Management Protocol (SNMP) [1]. It is an application-layer protocol that can exchange information between network devices. It defines commands and data representation to perform management tasks. Its advantage is simple and easy to use. But it is too simple to manage application behavior.

The Common Information Model (CIM) [2] defined by the Distributed Management Task Force (DMTF) is a powerful, object-oriented information model for management purpose. CIM is based on UML and is independent of any hardware architecture, operating system, and programming languages. It consists of a core model and many extension schemas. CIM is widely used by hardware and software vendors. However, applications have to implement several objects and methods to fulfill the requirements of CIM. It will increase development effort.

Java Management Extensions (JMX) [3] is used to manage in Java environment. It provides tools for building distributed, Web-based, modular and dynamic solutions for managing and monitoring devices, applications, and service-driven networks. By design, this standard is suitable for adapting legacy systems, implementing new management and monitoring solutions, and plugging into those of the future.

These industry standards all focus on management protocol and management console. It costs a lot of effort to implement those protocols when an application will be managed. More importantly, it is almost impossible to re-write all existing applications for management purpose. So, an automatic management solution is needed.

2.2 Objectives
Developing distributed applications is hard. It has to consider many problems in distributed environment. So, a simplified programming style is very important. Applying management capabilities to applications automatically is very attractive. Designers and developers can discard those management functions. Designers can focus only on business requirement and developers only need to write program for the requirement. It not only saves time to develop the application; simplified architecture also reduces error during developing.

3 Management Analysis
Analysis of manageable elements will understand what kind of resources should and could be managed. The management resources are classified as hardware resources, operating system resources, and application behavior.

3.1 Manageable Resources
Hardware resources include CPUs, memory, hard drives, network cards, etc. Those resources are physical entities with pre-defined interfaces that can be monitored and controlled. For example, operating systems can detect memory usage and the free space of hard drives.

Operating system resources include processes, threads, files, shared libraries, etc. Those resources are controlled by the operating system and help applications to develop functions more convenient.

The third one is application behavior. It performs the core functions to fulfill users’ needs. The kind of behavior is the hardest to control because users have many kinds of requirements. To manage application behavior, two aspects need considering:

1. Application logic. Applications are developed to solve specified problems. They process inputs and output the result. So, they should have statements, operations, and use existing applications. the complexity depends on function requirements. The part of program is usually unnecessary to monitor because there will be defined information for each situation.

2. Existing systems support. Some functions of applications may need functions from existing systems. For example, many applications need database function to retrieve data. It is hard to know the status of other systems because they are usually at different computers in distributed environment. For this reason, management agent will pay more attention to the usage of existing systems.

Based on the manageable resources and application behavior, the management agent should not only collect the information from the underlying operating system but also have to know what systems are used by the application. The most important is that the designers and developers will not need to develop
those functions on their own.

3.2 Hook Technology
To monitor and control those applications without modifying program code, adopting management functions into operating system is the best solution. It only needs to be configured so that it will get information from operating system. Operating system, of course, has everything about executed applications. However, how to get the information from operating system is the problem. Intercepting operating system application programming interface (API) is a solution. It can get control over a particular piece of code execution. It can alter the operating system’s behavior, applications, and third party products without having to modify their source code. It is hook technology.

Hook technology can be applied to several popular operating systems, such as Microsoft Windows 2000, XP, and Linux with Kernel 2.4 or above. Although different operating systems have different implementations, the same concept can still be applied. The hooks can spy application activities and provide the following advantages:

1. Monitoring API functions: hooks can monitor all API calls in applications. It can validate the input parameters and review the return value from API functions. For example, it can detect database error when application has called database API.
2. Controlling resources of operating system: it can limit resource used when a resource is used abnormally or reserved for other high priority functions. For example, hooks can limit memory usage when memory lacks.
3. Extending originally offered functionalities: hooks can insert actions for any applications. Application designers and developers can discard management functions. It can be extended after the application is installed. Besides, management elements can be reused when they have the same management features. It will save lots of developing time.

With hook technology, all applications can be monitored. Based on the analysis of application behaviors, hooks can track all necessary actions at application runtime. It can also setup application behaviors so that error could be prevented.

On both Windows and Linux platforms, we can inject system calls and redirect them to our management code. When applications call those functions, it will first call our management code. For example, if applications need a shared library to a specified function, hooks can intercept and invoke another code rather than the shared library. Finding the library to be hooked becomes the key to implement the system. For this reason, analysis of managed resources is important. Based on those resources, the corresponding shared library will be hooked and inserted the management functions.

4 System Implementation
We have implemented a system to apply the management mechanism to existing applications. Our implementation environment is as follows:

Operating systems: Microsoft Windows 2000 and Redhat Linux 9. Both are installed on Intel Pentium 4 2.4 GHz with 512 MB RAM.

Program Languages: C/C++ and Java. C/C++ is responsible for exchanging message with operating systems. Java gets information from C/C++ and constructs management user interface.

The high level system architecture is showed as Fig. 1. The server will show the status of the distributed environment and control its state by sending command. The client will collect information and perform the command from the server. In this implementation, we focus on the management mechanism on the client. The server is implemented to collect and evaluate the solution.

4.1 Management Functions
Management functions of the system can be considered as three aspects: monitor, alert and interception. The functions of monitor include: (1) Connection: the request and response of connection in a single computer. (2) CPU and memory utilization. (3) Any actions on file. For example, delete or create a file. (4) Sending and receiving e-mail. (5) Network packets. (6) The status of operating system processes.
The alert functions will notify the related receivers to process the alert. It can do the following jobs: (1) Sending message to manager of the system. (2) Preventing the error occurred. (3) Stopping the process if it will hurt the whole system. (4) Logging the alert.

Finally, the interception functions include: (1) Preventing the network connection. (2) Preventing file creation or deletion. (3) Preventing sending/receiving e-mail. (4) Shutting down network. (5) Preventing the process creation.

Based on the three aspects, every function on the system will be monitored and controlled. Although the usage of CPU and memory can be managed already, e-mail functions are not. In the section 4.4, to manage e-mail functions without modifying email client will be presented.

### 4.2 Client Design Model

To manage client resources, a suitable model to build the management program is built, showed as Fig. 2. The management program, named as prob, is installed on client and communicates with the server. The design of probes is based on UML and follows many design patterns. It will make it easy to integrate with industry management infrastructure.

**Probes** are implemented on two different platforms: Linux and Windows. However, they all follow the same design and have the same architecture. Each probe will design its own Listener. Whenever an event is fired, they will send messages to the server.

### 4.3 Managed Resources

Table 1 shows the actions for different resources on different platforms. Some resources are not only monitored but also controlled. In the Linux environment, it cannot prevent an outgoing network message. It can only monitor it. However, on the Windows platform, we can stop sending a network packet.

When applications use other systems, like database, those systems should provide shared library. By analysis of the library, the management functions can be defined. Some may be monitored and others can be intercepted.

### 4.4 Client Probes

To manage those resources listed in Table 1, several probes had been implemented. Most of them can be controlled from operating system directly. Operating system provides APIs to let programmers get system status. Both of Windows and Linux have similar features. Connection, CPU, file, network, process, and RAM can be controlled directly. They are also independent of application logic.

To control the sending and receiving of email, we use hook technology to process network request. Although sending and receiving mails are through standard protocol, such as POP3 or SMTP, Network Probe can monitor network information. However, Network Probe cannot compose entire mail by network packets. It cannot know the program to which each packet belonged. In other words, Network Probe cannot classify packets that application need. Besides, intercepting sending and receiving mails also add more extra functions to the original mail.

### Table 1: Actions for Each Managed Resources

<table>
<thead>
<tr>
<th>Resource</th>
<th>Win32</th>
<th>Linux</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection</td>
<td>monitor</td>
<td>interruption</td>
</tr>
<tr>
<td>CPU</td>
<td>monitor</td>
<td>monitor</td>
</tr>
<tr>
<td>File</td>
<td>monitor</td>
<td>interruption</td>
</tr>
<tr>
<td>Mail</td>
<td>monitor</td>
<td>interruption</td>
</tr>
<tr>
<td>Network</td>
<td>monitor</td>
<td>interruption</td>
</tr>
<tr>
<td>Process</td>
<td>monitor</td>
<td>interruption</td>
</tr>
<tr>
<td>RAM</td>
<td>monitor</td>
<td></td>
</tr>
</tbody>
</table>
clients if it cannot specify what kinds of mails can be accepted. The management mechanism can be adding by configuring Mail Probe so that mail client will receive and send the legal content.

To archive the functions on Windows platform, a wsock_hook.dll is inserted between TCP/IP transport layer and the mail client, shown in Fig. 3. The behavior of original applications is shown as left side. The mail client uses network and TCP/IP transport by WS2_32.dll. The right side is our solution to control mail. Its implementation is as follows:

1. Install our wsock_hook.dll into system by using WSCInstallProvider() and redirect the socket call of AF_INET + IPPROTO_IP + IPPROTO_TCP to the wsock_hook.dll
2. Rewrite WSPConnect() in wsock_hook.dll to replace the original one.
3. If the connection is SMTP/POP3 port, wsock_hook.dll will analyze the connection and get the mail content.
4. Based on the pre-defined rules, wsock_hook.dll will reject or pass the mail.

Hooks on RedHat Linux 9 has similar solution. Its actions are as follows:
1. Intercept the system call about sock operation. Linux kernel has sys_call_table[] and can register our functions into the table. Then, all sock operations will be redirected to the registered function.
2. Rewrite socketall() function and check PF_INET protocol family. If the connection is SMTP/POP3 protocol, the mail analysis will start, the same as Windows platform.

Although the original mail client cannot choose to receive what kind of mail, the mail filter and monitor mechanism are built by the hook technology. The same approach can also be applied to monitor database operations or other existing systems without modifying any systems.

4.5 Performance Impacts
Management system should not affect other applications. It should not occupy too much CPU and memory resources. On the other hand, its response time should be short so that the server can get up-to-date information.

After monitoring the system management system, each management message can be sent to the server almost real-time in the same sub-network when many probes have events simultaneously. Normally, the CPU usage is low: around 2% to 30% on Intel Pentium4 2.4 GHz. It depends on the number of messages to be sent. If sending 10 messages per second, it will use 2% CPU. Memory has similar situation. It will use 50 Mbytes memory. However, most of them are created by Java. The monitor program only uses 10 Mbytes with sending 10 messages per second. For a distributed environment, the required resources are not huge.

5 Conclusion
In this paper, we have proposed a solution to adopt management mechanism to existing applications. It can monitor and control the client status by hook technology. Applications can be not only monitored but also less error-prone. We also have completed management architecture for the client side. All kinds of managed resources can be used. Following the system architecture, the developing effort will be saved and the system complexity will become lower.

In the current system, we develop a server and native protocol. It cannot communicate with existing management products. The next step is to follow the specifications of CIM. Our system is designed by UML and can be integrated with CIM easily. DMTF has defined CIM Diagnostic Specification [4]. Besides, we will also find more sample functions to enrich the management capabilities.

Acknowledgments
This work was partially supported by National Science Council grant NSC95-2752-E-009-PAE: advanced technologies and applications for next generation information networks, grant NSC94-2213-E-009-026: a research on next generation massive multiplayer virtual environment platform, and grant NSC94-2520-S-009-004.

References:


