Embedded System Management

Jin-Ho Ka
Pantec & Curitel

Jai-Hoon Kim*, Won-Sik Yoon
College of Information and Communications
Ajou University, Suwon, Korea

Daein Jeong
College of Information and Industrial Engineering
Hankuk University of Foreign Studies, Yongin, Korea

ABSTRACT

This paper proposes a frame work for embedded system management. The management system can automatically monitor and maintain the Internet connected embedded-systems such as sensor devices, set-top box, web-pad, information appliances, PDA, etc. Users can easily diagnose system state, resolve system problems, maintain and update applications, and back up user information using the management system. We implement prototype for the management system on embedded Linux system and High -available Linux system.

***Keywords**: Embedded system, Management, Monitoring, Software rejuvenation, Software update*

1. INTRODUCTION

Besides PC, many users make use of Internet-connected devices in which small computer systems are required. These kinds of Internet-connected embedded systems such as smart home appliance, web-pad, TV set-top box, cellular-phone, sensor devices and PDA tend to offer users more convenience than traditional devices. But by inheriting nature of complex computer systems, they may need some management from time to time. Monitoring system state, updating software and even more enhancing system performance and preventing system failures are needed. Unlike PCs having standard architecture and software, it might be very hard or impossible for users to do maintain embedded systems. Thus we propose a framework for the Internet-connected embedded system management that automatically monitors and maintains the embedded-systems so that we can maintain and enhance the computing resource effectively. In prototype for management system is implemented on embedded Linux system kit and High-Available Linux system.

There are many system monitoring software for PCs, workstations and network devices such as BigBrother[1], MRTG[2], and MON[3]. System failure preventing software for server system have been developed such as IBM Director software [4]. Some operating system and application software offer the automatic software update via network for PCs and servers. But, there are no notable products that perform monitoring system, preventing system failures and updating software for the embedded systems like information appliance.

Exhaustion in the availability of system resources and numerical error leads to software aging[5]. Software aging can cause system failure and performance degradation. Many embedded systems have to operate long time without stop and need to high availability. These kinds of embedded system can be suffered from software aging. Software rejuvenation[6,7] is a proactive technique for fault management by cleaning up the system state to avoid or reduce the occurrence of severe crash failures[8]. Software aging in the target systems can be detected by monitoring the system state. We can restore system to initial or clean state using software rejuvenation. Recently IBM developed Director software rejuvenation utility for eServer xSeries with Duke University [9]. As a part of server management tool, IBM Director offers system monitoring and software rejuvenation.

*Corresponding author. E-mail: jaikim@ajou.ac.kr
Manuscript received Jan 24, 2005 ; accepted Feb 2, 2005

2. EMBEDDED SYSTEM MANAGEMENT FRAMEWORK

We propose the following schemes for embedded system management: (1) remote monitoring the Internet-connected embedded systems for easy trouble-shooting and performance improvement, (2) reducing the maintenance cost by automatic software update, maintaining the user data on the embedded systems, and providing the useful information, (3) preventing system failure by the software aging, and (4) developing the high available management server supporting the embedded systems, which offers embedded system information and controls embedded systems.

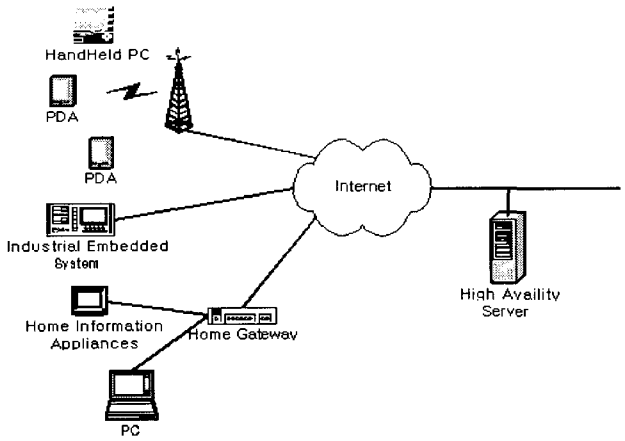


Fig. 1. System Configuration of the Embedded-System Management

Fig. 1 shows our system configuration. for embedded system management. The proposed system offers following features: monitoring system state, preventing system failure, maintaining user/application data, and updating software for embedded systems.

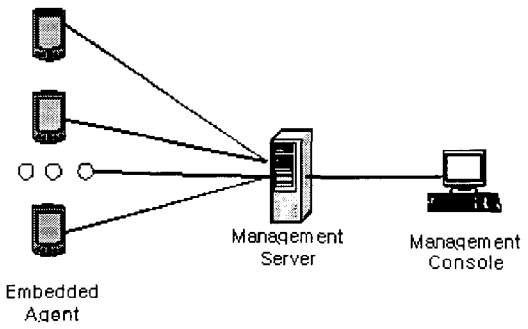


Fig. 2. System Framework

The embedded system management consists of three parts: (1) the embedded agent, (2) the management server, and (3) the management console as depicted in Fig. 2. We implement prototype for the management system on our test bed consisting of Embedded Linux system and High-Available Linux system.

2.1 Embedded system monitoring

The embedded system monitoring is executed via Internet connection. When a management server requests embedded system state, the embedded systems gather system state information then send back system information to the management server. Because the embedded system has limited system resources, the activity of monitoring on embedded system may reduce system performance. Therefore the embedded agent that performs monitoring should be implemented small and simple. In our implementation, embedded system monitoring deals with CPU, memory, disk, and network usage.

2.2 Software rejuvenation

Software rejuvenation method is used for preventing system failures caused by software aging. There are two ways to achieve software rejuvenation: (1) when software aging is detected by monitoring system state, rejuvenation is performed, or alternatively rejuvenation is performed periodically by pre-defined time table. Software rejuvenation is performed by restarting or cleaning system or application. Then aged system or application restarts from an initial state or the last saved state. In our implementation for software rejuvenation is executed in system level. Software rejuvenation is performed at detection of software aging or pre-defined time.

3. IMPLEMENTATION FOR EMBEDDED SYSTEM MANAGEMENT

Test bed for our prototyping consists of Embedded Linux system as a embedded system and High-Available Linux system as a management server. The test bed is described in Table 1.

Table 1. Test Bed System Setup

System Types	Hardware specification	O S	Assumed role
Embedded system	• Embedded Board - Intel StrongARM processor - 16MB Memory, 16MB Flash	• Tynux (Embedded Linux)	• PDA • Home Information appliance • Set-top box
Management server	• PC - Intel Pentium 3	• RedHat Linux 7.0	• Home PC • Web pad
	• PC - Intel Pentium 3	• RedHat Linux 7.1	• High available server
Management console	• PC - Intel Pentium 3	• RedHat Linux 7.1 • Windows 2000	• PC

The test bed for embedded system is Tynux box that is an embedded system development board from Palmpalm tech[10]. The Tynux box is equipped with StrongARM processor which is widely used for recent PDA systems and operating system for Tynux box is Tynux embedded Linux. The operating system of Tynux box is a small size embedded Linux developed for portable Internet-connected devices by Palmpalm tech. The test beds for management server and management console platform are Intel Pentium III PCs with Redhat Linux 7.1. The management server is on typical Linux system but in future implementation; management server will be high available Linux system. The management console program is able to run on both Linux and Windows OS. We has developed management console program on Linux system with SUN Java SDK 1.3.

3.1 Embedded Agent

The embedded agent resides on embedded systems as an application layer and it monitors system state, restarting the applications or the operating system to recover from failures, and updating software by the management server. The embedded agent can not store large data in local embedded system but in management server. TCP or UDP protocol is used for network connection via Internet. On embedded system, the embedded agent is executed as daemon or invoked by inetd daemon.

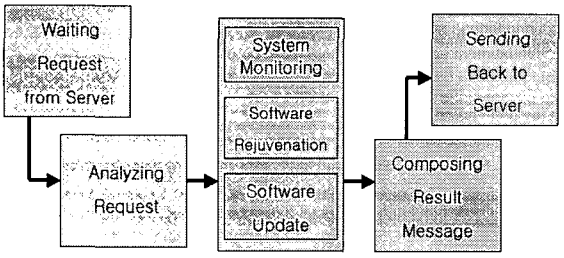


Fig. 3. The Embedded Agent Functions

Fig. 3 shows a flow chart of functions performed in an embedded agent application. When the agent receives monitoring request from a management server, it gathers system information by reading proc file or calling Linux system call and report monitoring results to the management server. When the agent receives rejuvenation request, it decides whether to perform system level rejuvenation or application level rejuvenation, then it restarts all application or only a specified application on system, and finally sends rejuvenation result message to the management server. The software update occurs when the agent receives update request. When the agent receives updated software from server, it restarts system as well as performs software rejuvenation. In our test bed, updating software has simply implemented replacing the Linux kernel image file and the file system image files. Then embedded agent restart system with updated kernel and file system.

3.2 Embedded Management Server

The management program on High-Available server holds all needed data as XML data format for embedded system management. The management program schedules monitoring system state and restarting application/operating system, which is a method of software rejuvenation to refresh the system state to prevent the system from failures, and updates software by preset plans or automated algorithms. The management server also offers some additional facilities such as storage space for the embedded systems suffering from lack of disk space. We have implemented system monitoring and rejuvenation functions and other additional functions will be implemented by the second quarter of 2002.

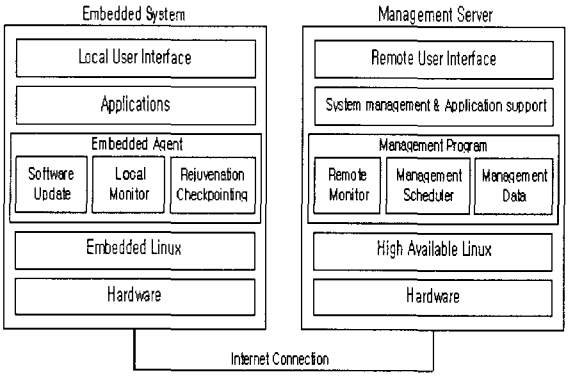


Fig. 4. Software Configuration

Fig. 4 shows software configuration of the management server and the embedded agent. The management data consists of embedded system information, monitoring schedule, rejuvenation scheme, software update plan and monitoring/rejuvenation result. Using XML for storing data has a number of advantages comparing with database or proprietary data format. With XML we can easily transform XML data to other data format and manipulate data. The management server application written in JAVA uses Apache Xerces parser library with SAX, DOM from W3C for manipulating XML data. Fig. 5 and 6 are XML data examples of our implementation.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE RESULT SYSTEM "results.dtd">
<Result>
<System>
        <Serial>1</Serial>
                <Monitor>
                <IP>202.30.0.11</IP>
                <Time>
                        <Year>2002</Year>
                        <Month>1</Month>
                        <Day>1</Day>
                        <Hour>12</Hour>
                        <Min>0</Min>
                </Time>
                <CPU>20</CPU>
```

```
                    <Mem>60</Mem>
                    <Disk>10</Disk>
            </Monitor>
</System>
</Result>
```

Fig. 5. Example of Result.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE PLAN SYSTEM "plan.dtd">
<Plan>
<System>
            <Serial>1</Serial>
            <Register>yes</Register>
            <Name>Tynux1</Name>
            <Address>202.30.0.11</Address>
            <Info>
                    <Type>Test Board</Type>
                    <Etc>Test</Etc>
            <Monitor>
                    <Type>Active</Type>
                    <Period>5</Period>
            </Monitor>
</System>
</Plan>
```

Fig. 6. Example of Plan.xml

The simplified flow chart for a management server application is depicted in Fig. 7.

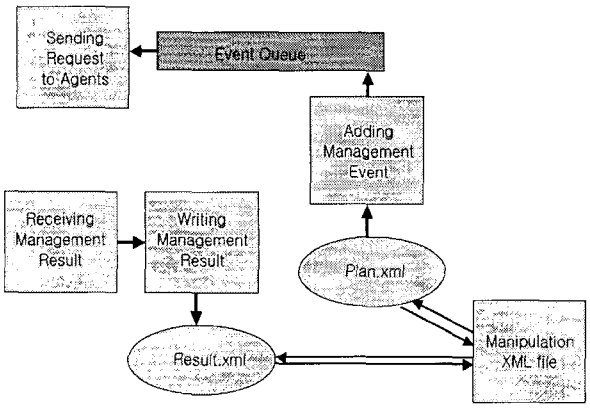


Fig. 7. Management Server Functions

In the event of monitoring and rejuvenation for embedded systems, "Plan.xml" holds information and management scheme for each target system. The management server application reads "Plan.xml" file then composes management events: which management job will perform, and when the job will perform. The management event is added to the event queue and the server send management request when the time expired in management event. After embedded system processes management request, a management server receives result message and write it in "Result.xml". Monitoring events in embedded system occur periodically with predefined interval for each system, typically 5 minutes, or at a specified time of the day, week, and month. Software rejuvenation happens in two ways. The administrator can specify the rejuvenation time, which is periodic or one shot. The rejuvenation periods should depend on embedded system characteristic. The second way of software rejuvenation is based on monitoring results. In our implementation, administrator set threshold value of percentage of memory usage or CPU usage. When monitored value of embedded system exceeds this threshold, the rejuvenation performs automatically. Software update is performed in similar way of monitoring and rejuvenation but transferring software is needed. Based on information of embedded systems from "Plan.xml", management server prepares a proper kernel image and a file system image which holds application code and data, respectively. Then it sends a software update notification message and starts transmitting the kernel image and the file system image. After embedded agent receives the image files and restarts system, it send back a software update confirmation message to the embedded management server. The embedded management server receives the message that the embedded system finished a software update, then it checks the updated software version information of the embedded system and writes a software update result log into "Result.xml".

3.3 Embedded Console

The administrator can obtain all information about the embedded systems and manipulates management plans and chooses algorithms on the remote management console. The facilities provided by the console application are shown in Table 2.

Table 2. The Functions of Management Console

Function types	View	Input/Edit
Basic	• Embedded system information	• Embedded system information
Monitoring	• Embedded system state	• Monitoring period • Monitoring items
Rejuvenation	• Software rejuvenation log	• Rejuvenation scheme • Rejuvenation method
Update	• Software update log	• Update scheme • Software data

The administrator can perform the following transactions on the console: (1) add and remove target embedded systems, (2) determine monitoring period and rejuvenation policy, (3) obtain information about embedded system monitoring and

rejuvenation results. The embedded console application is written in JAVA so that user can install console program on Windows as well as Linux.

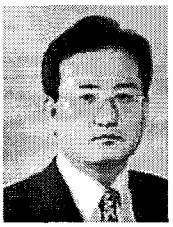
4. CONCLUSION

Proposed management schemes for embedded system integrate system management functions such as monitoring system state, software rejuvenation and software update. The aim of this research is providing efficiencies and value-added to Internet-connected information devices and smart appliances. Our research and proposed embedded-system management system are able to improve the Internet-connected embedded system performances without annoying continuous attention for embedded systems.

REFERENCES

[1] Big Brother website, http://bb4.com/.
[2] MRTG website,
[3] http://people.ee.ethz.ch/~oetiker/webtools/mrtg/.
[4] MON website, http://www.kernel.org/software/mon/.
[5] "IBM Director Software Rejuvenation," IBM xSeries and Netfinity Technical Information white paper, Jan. 2001.S. Grag, A. Moorsel, K. Vaidyanathan and K. S. Trivedi," A Methodology for Detection and Estimation of Software Aging," International Symposium on Software Reliability Engineering, November 1998.
[6] K. S. Trivedi, K. Vaidyanathan and K. Goseva-Popstojanova, "Modeling and Analysis of Software Aging and Rejuvenation," IEEE Annual Simulation Symposium, April 2000.
[7] S. Garg, A. Puliafito, M. Telek and K. S. Trivedi, "On the Analysis of Software Rejuvenation Policies," Annual Conference on Computer Assurance, June 1997.
[8] Soft rejuvenation homepage, http://www.software-rejuvenation.com/.
[9] V. Castelli, R. E. Harper, P. Heidelberger, S. W. Hunter, K. S. Trivedi, K. Vaidyanathan and W. P. Zeggert, "Proactive management of software aging," IBM J. REs & DEV., Vol. 45, No. 2, March 2001, pp. 311-332.
[10] Palmpalm tech site, http://www.palmpalm.co.kr.

Jin-Ho Ka
He received the B.S., M.S in Computer Engineering from Ajou University, Korea in 2002, 2004, respectively. He is currently working for Pantec&Curitel. His main research interests include embedded systems and mobile computing.



Jai-Hoon Kim
He received the B.S. degree in Control and Instrumentation Engineering, Seoul National University, Seoul, South Korea, in 1984, M.S. degree in Computer Science, Indiana University, Bloomington, IN, U.S.A., in 1993, and his Ph.D. degree in Computer Science, Texas A&M University, College Station, TX., U.S.A., in 1997. He is currently an associate professor of the Information and Communication department at Ajou University, South Korea. His research interests include Distributed Systems, Real-Time Systems, Mobile Computing and Fault-Tolerant Systems.

Won-Sik Yoon
He graduated with the B.S. degree in Control and Instrumentation Engineering from Seoul National University in 1984. He received the M.S. and the Ph. D. degrees from KAIST in 1986 and 1991 respectively. From 1986 to 1994, he worked with Goldstar Electrical Company and LG Innotek, South Korea. He served as an invited professor at the University of Victoria from 1995 to 1996. In 2000 and 2001, he worked as the CTO at Contela Inc., South Korea. Since 1994, he has been with the Department of Electrical Engineering at Ajou University, South Korea, where he is a Professor. His research interests include ubiquitous systems and networks as well as smart homes, sensor and ad hoc networks, and telematics.

Daein Jeong
He received the B.S. and M.S. degree from Seoul National University, Seoul, Korea, both in control and instrumentation engineering in 1984 and 1986, respectively, and Ph.D. degree in electrical engineering from Polytechnic University, Brooklyn, New York, in 1997. From 1987 to 1999, he was with Korea Telecom, Korea, where he researched the next generation Internet architecture. He is currently an Associate Professor in the School of Electronics and Information Engineering at Hankuk University of Foreign Studies. His research interests are in the field of network performance, Internet QoS, and embedded system such as sensor networks.