BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **13**, No 4

Sofia • 2013 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2013-0057

A Mobile Agent-Based System for Server Resource Monitoring

Zhixin Tie

School of Information Science and Technology, Zhejiang Sci-Tech University, Hangzhou, P. R. China
Email: tiezx@zstu.edu.cn

Abstract: *Mobile agent technology has become an important approach for the design and development of distributed systems. Currently, there is little research regarding the efficiency of mobile agent-based monitoring of the server resource. Based on the Mobile-C library, a mobile agent-based system called Mobile Agent-Based Server Resource Monitoring System* (*MABSRMS*) *is presented. In MABSRMS mobile agents can call low level functions in binary dynamic or static libraries, and thus can monitor server resource conveniently and efficiently. The experiment was conducted in a university computer center with hundreds of computer workstations and 15 server machines. The experiment uses the MABSRMS to detect system resources, such as available hard disk space, CPU usage and main memory usage. The experiment shows that the mobile agent-based monitoring system is a practical way to monitor server resources in large scale distributed computer centers.*

Keywords: *Mobile-C library, mobile agents, computer laboratories, server resource monitoring.*

1. Introduction

A server is a physical computer or a computer hardware system that makes available files, database, printing, email, communications or other services to client terminals/stations with access to the network where the server is located. Depending on the computing service that it offers, it could be a database server, a file server, a mail server, a print server, a web server, a gaming server, or another kind of a server. Servers are available in different sizes, shapes and varieties. Servers may be distributed throughout a network or they may be concentrated in centralized data centers. The computing services that run on the servers are the core business of companies or organizations, thus it is very important to keep the servers working and keep the computing services running smoothly. This is also the key work of system administrators. In time the users in the services will be increased, users tend to demand faster, more convenient services, thus in most of the cases, the servers' load is getting heavier and heavier. Because servers' resource is limited, it is very important for system administrators to monitor server's performance. Server's performance data can be applied to the performance management, can be used to detect faults in the fault management, can be used to decide how to adjust servers' configuration, and also to regulate the bill in the billing management.

A good server resource monitoring system must have the following characteristics:

(1) To work automatically and work around the clock after deployed.
(2) To be deployed easily, quickly and dynamically.
(3) It must have a small footprint.
(4) It must be an operating system-independent system.

To monitor the status of smaller numbers of computer systems, an individual can switch among computer screens of each of the computer systems, querying for a status as they switch among the screens, but to monitor dozens or hundreds of computer systems would require a larger effort 1. Several centralized architecture based server resource monitoring systems have been presented 3, 4. Since a large amount of the management data must be transmitted to the centralized management station for processing, these systems suffer from scalability and flexibility problems 5.

Mobile agent technology has become a powerful technology for designing and implementing various applications in a computer networking environment. The use of mobile agent technology for network management 5-10, a distributed sensor network [11-13 and intrusion detection [14-16 has been presented by several researchers in recent years. However, there is little research regarding the efficiency of the mobile agent-based monitoring of a server resource.

In this paper we present a mobile agent-based system called Mobile Agent Based Server Resource Monitoring System (MABSRMS), which is based on the Mobile-C library to monitor server resource at public computer centers. In MABSRMS, an agency starts on each of the servers during the boot time. Monitoring agents are dynamically sent to a group of or to all of the servers from a monitoring server when monitoring tasks are needed. The monitoring results are transferred from the servers to the monitoring server by the agent itself.

The rest of the paper is organized as follows. Section 2 presents a Mobile Agent Based Server Resource Monitoring System (MABSRMS) based on the Mobile-C library. Section 3 demonstrates an application of the system. Finally, conclusions are drawn in Section 4.

2. Mobile agent-based server resource monitoring system

2.1. Mobile-C library

Mobile-C library [17-19] was originally developed as a stand-alone, FIPA compliant mobile agent system. It uses an embeddable C/C++ interpreter, Ch [20-22, as the Agent Execution Engine (AEE) to support the interpretive execution of C/C++ mobile agent codes.

The major components of the Mobile-C library include Agency, Agent Management System (AMS), Agent Communication Channel (ACC), Agent Security Manager (ASM), Directory Facilitator (DF), Agent Execution Engine (AEE), Agent, Synchronization, and Miscellaneous APIs. The Mobile-C library has extended most of the API functions from the host program space to the mobile agent space. The mobile agent space APIs allows a mobile agent to interact with an agency, different modules of an agency, and other agents. The right part of Fig. 1 shows how the mobile agent code interfaces with Mobile-C library. When the function mc_Function is called in mobile agent code, Ch searches the corresponding interface function MC_Function_chdl in the Mobile-C library, and passes arguments to it by calling the function. Subsequently, the interface function MC_Function_chdl invokes the target function MC_Function, and passes the return value back to the mobile agent space.

Since the AEE of the Mobile-C is based on Ch and Ch is capable of calling functions in binary static and dynamic libraries without recompilation [23-25], all existing binary static and dynamic C libraries and modules can be used as a Mobile-C agent code. For example, functions in OpenGL and XML libraries can be called from Ch directly [26, 27], Ch communicates with the functions in binary libraries using a dynamically loaded library, as shown in the left part of Fig 1.

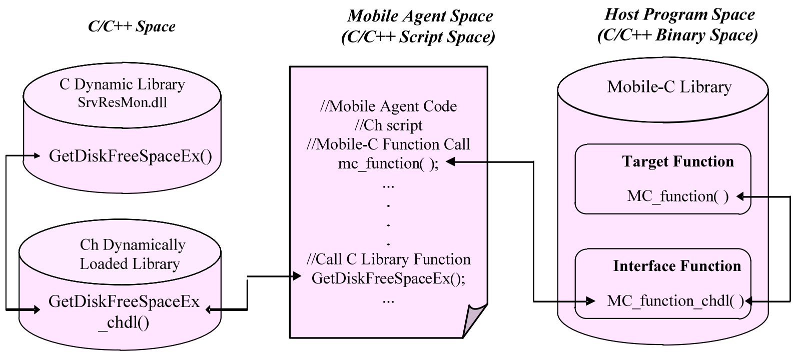


Fig. 1. Interface of a mobile agent code with Mobile-C library and a common C dynamic library

The example shows how a mobile agent code invokes the functions in the C dynamic library, called SrvResMon.dll in Windows (or in the shared object called SrvResMon.so in LINUX), which is developed to get the CPU information and the CPU utilization information, the available and used hard disk space, the physical memory space and memory utilization information, and other connected information from the operation systems.

2.2. Architecture of MABSRMS

The architecture of MABSRMS is shown in Fig. 2. Each monitored server, known as a client node, runs a monitoring program that encompasses a Mobile-C agency. Multiple mobile agents can run in the agency at the same time. When the agency receives a mobile agent that is sent from the monitoring server, an AEE will be automatically created to run the agent code immediately. If the monitoring server needs some results, the result will be carried by the original mobile agent back to the monitoring server automatically after the agent is executed. If the agent has a migration task, it will migrate to its next destination automatically after its task on the current client node is completed. When the agent is running at the client node, it can access all the client node resources via the method mentioned in Section 2.1.

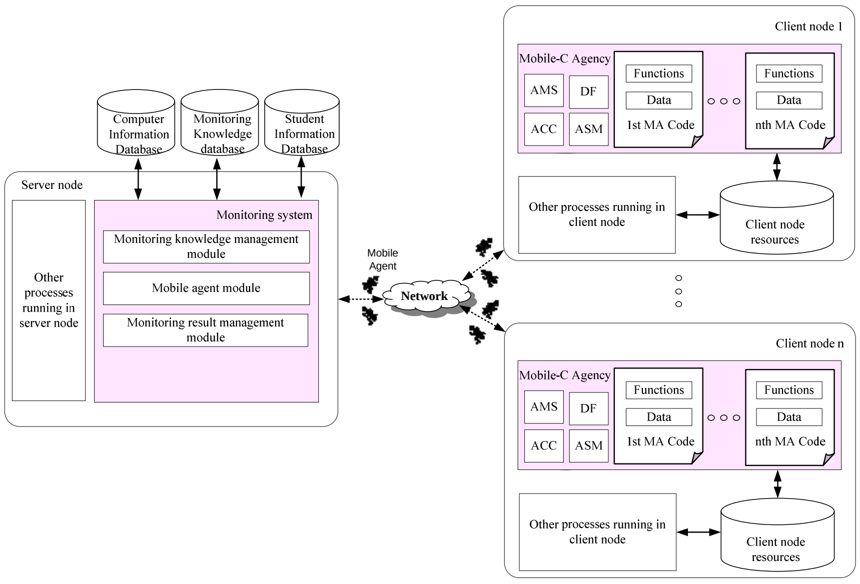

Computer Information Database
Monitoring Knowledge database
Student Information Database
Server node
Other processes running in server node
Monitoring system
Monitoring knowledge management module
Mobile agent module
Monitoring result management module
Mobile Agent
Network
Client node 1
Mobile-C Agency
AMS
DF
ACC
ASM
Functions
Data
1st MA Code
nth MA Code
Other processes running in client node
Client node resources
Client node n

Fig. 2. The architecture of MCBAS

There is a monitoring server program running on the monitoring server. The monitoring server program includes the Monitoring Knowledge Management module (MKM), the Mobile Agent Module (MAM), and the Monitoring Result Management module (MRM).

The functions of MKM are to manage the monitoring knowledge. For example, adding/removing a server to/from the system, setting alarm thresholds of CPU usage for a monitored server.

The functions of MAM are to create mobile agents, manage mobile agents' code, send mobile agents to client nodes, and receive the monitoring result data by inspecting the returning mobile agents and also store data in the monitoring knowledge database. For example, if a new server resource monitoring algorithm has been developed, it can be stored in the monitoring knowledge database through this module. A new mobile agent powered by the new monitoring algorithm can be created and sent to the client nodes for testing or getting the designated information of the client nodes.

The MRM is used to analyze the monitoring data and generate reports. By accessing the server information database and monitoring knowledge database, the report about the computing resource usage of the servers will be automatically generated and emailed to the designated recipients daily.

2.3. Mobile-C agent

The Mobile-C agent is composed of a mobile agent code in Ch which is an embeddable C/C++ interpreter, as mentioned in Section 2.1, encapsulated in XML format, as shown in Fig. 3. Each Mobile-C agent has several attributes, including the name, owner, home address, and tasks of the agent. Each agent task has attributes, such as the ordinal number of the task, the return variable, completeness, persistence, execution host of the task, and agent code of the task. The persistence attribute can be enabled to create an agent that will not be removed from the agency after the agent code is executed. In this way the variables and functions in the agent code can still be accessed later on. The Mobile-C agents can have sophisticated dynamically generated task lists and move around a network autonomously. For a detailed format, please refer to [18, 19].

```
<MOBILE_AGENT>
  <AGENT_DATA>
   <NAME> HDUA </NAME>
   <OWNER>ZSTU</OWNER>
   <HOME>localhost:5050</HOME>
   <TASKS task="3" num="0">
    <TASK num="0" complete="0" Server="10.16.23.2:5070" persistent="1" return="fHard" />
    < TASK num="1" complete="0" Server="10.16.23.3:5070" persistent="1" return="fHard" />
    < TASK num="2" complete="0" Server="10.16.23.4:5070" persistent="1" return="fHard" />
    <AGENT_CODE>
     <![CDATA[
      #include <stdio.h>
      #include <Windows.h>
      double  fHard;
      int main()
      {
            char sDriver1[10]="c:";
            ULARGE_INTEGER AvailableToCaller, Disk, Free;
            if (GetDiskFreeSpaceEx(sDriver1,&AvailableToCaller,&Disk, &Free)){
                  fHard = (double)((long double)Free.QuadPart/1024/1024);
            }
            else{
                  fHard = 0;
            }
            return 0;
      }
     ]]>
    </AGENT_CODE>
   </TASKS>
  </AGENT_DATA>
 </MOBILE_AGENT>
 </MESSAGE>
</MOBILEC_MESSAGE>
```

Fig. 3. The Hard Disk Usage agent on the Windows platform

3. Monitoring server machine resource

This application uses MABSRMS framework that is introduced in Section 2 to monitor server machines' resource. The monitoring server sends mobile agents to the monitored servers and gets servers' workload performance data, such as the information of the main memory, CPU usage, and hard drive space of the monitored server. Using servers' workload performance data, the bottlenecks of the servers can be found, workloads in the servers can be balanced, and the hardware of the servers can be properly upgraded.

3.1. Monitoring environment introduction

In this application MABSRMS is used to monitor the resource of server machines of a university computer center. There are 15 servers and hundreds of computer workstations in the computer center. In the server room, all 15 servers are installed in cabinet racks. Different applications are installed on the servers, such as DBMS, FTP Service, Mail Service, and variety of educational software. Traditionally, server's resources, such as CPU usage, main memory usage, and available hard disk space, are monitored by manual inspections. Because there is only one terminal which we can use to interact with the servers, we know little about the servers' workload performance. Thus, a program for continuously monitoring the server resource consumption is strongly desired.

3.2. Monitoring the server machine resource agents design

There are many kinds of resources in the server machines, including hardware, software, network, etc. In this monitoring example, we are only concerned about the usage of the main memories, CPUs and the hard drive, because they are some of the most crucial factors that affect the server performance. Three mobile agents are designed to get the information.

(1) The Hard Disk Usage Agent (HDUA). The hard disk usage agent is used to get the total disk capacity and the currently available disk space of each hard disk of a server. As shown in Fig. 3, a HDUA can be a single-task or multi-task mobile agent depending on how many tasks are configured in the agent. When a HDUA is created on the monitoring server, it will migrate to the monitored servers based on its tasks and get the desired information back to the monitoring server. As shown in Figs. 1 and 3, the function GetDiskFreeSpaceEx in the HDUA calls the function GetDiskFreeSpaceEx_chdl in Ch dynamic load library, and the function GetDiskFreeSpaceEx_chdl invokes the function GetDiskFreeSpaceEx which is an API function on the Windows platform. This API function retrieves information about the amount of space that is available on a disk volume, which is the total amount of space and the total amount of free space available to the user that is associated with the calling thread 27. For Linux platform, the HDUA can invoke the functions *statfs* or *statvfs* using the same method.

(2) The CPU Usage Agent (CPUUA). The task of the CPU usage agent is getting the usage of each CPU and the total CPU usage of a server. The agent code and the working mechanism of the CPUUA are similar to the HDUA. For the Windows platform, the CPUUA can invoke the Window API function GetSystemTimes which returns the idle time, kernel time, and user time 29. Ejor gives some details in 30. As shown in Fig. 4, call the function GetSystemTimes to get the CPU's idle time, kernel time, and user time at time t. Given times $t1$ and $t2$, the CPU usage can be calculated as follows:

(1) $\text{User\_time}=\text{UserTime}(t_2)-\text{UserTime}(t_1)$,

(2) $\text{Kernel\_time}=\text{KernelTime}(t_2)-\text{KernelTime}(t_1)$,

(3) $\text{Idle\_time}=\text{IdleTime}(t_2)-\text{IdleTime}(t_1)$,

(4) $\text{Total\_time}=\text{User\_time}+\text{Kernel\_time}$,

(5) Cpu_usage=(Total_time-Idle_time)/Total_time,

where $\text{UserTime}(t)$, $\text{KernelTime}(t)$ and $\text{IdleTime}(t)$ are the CPU's user time, kernel time and idle time at time t respectively, User_time, Kernel_time and Idle_time are the CPU's user time, kernel time and idle time from time t_1 to time t_2 respectively, Total_time and Cpu_usage are the CPU's total time and the CPU usage from time t_1 to time t_2 respectively.

```
FILETIME idleTime;
FILETIME kernelTime;
FILETIME userTime;
BOOL res = GetSystemTimes( &idleTime, &kernelTime, &userTime );
```

Fig. 4. Using the Windows API function GetSystemTimes to get the CPU's user time, kernel time and idle time

For a Linux platform, the CPUUA can use the /proc/stat file which includes various pieces of information about kernel activity 31. As shown in Fig. 5, the very first "cpu" line aggregates the numbers in all other "cpuN" lines. These numbers identify the amount of time the CPU has spent performing different kinds of work. Time units are in USER_HZ or Jiffies (typically hundredths of a second). The meanings of the columns of the very first "cpu" line are as follows, from left to right:

- user: Normal processes executing in user mode;
- nice: Niced processes executing in user mode;
- system: Processes executing in kernel mode;
- idle: Twiddling thumbs;
- iowait: Waiting for I/O to complete;
- irq: Servicing interrupts;
- softirq: Servicing softirqs.

Thus, the total CPU time and CPU busy time at time t can be calculated as follows:

(6) $$\text{Total\_cpu}(t) = \text{user}(t) + \text{nice}(t) + \text{system}(t) + \text{idle}(t),$$

(7) $$\text{Cpu\_busy}(t) = \text{user}(t) + \text{nice}(t) + \text{system}(t).$$

Given times t_1 and t_2, the CPU usage can be calculated as follows:

(8) $$\text{Cpu\_usage} = \frac{\text{Cpu\_busy}(t_2) - \text{Cpu\_busy}(t_1)}{\text{Total\_cpu}(t_2) - \text{Total\_cpu}(t_1)}.$$

```
> cat /proc/stat
cpu  2255 34 2290 22625563 6290 127 456
cpu0 1132 34 1441 11311718 3675 127 438
cpu1 1123 0 849 11313845 2614 0 18
…
```

Fig. 5. The information in the proc/stat file

(3) The Main Memory Usage Agent (MMUA). The main memory usage agent is designed to get the total physical memories and the currently available memories of a server. The agent code and the working mechanism of the MMUA are similar to HDUA. For Windows platform, the MMUA can invoke the Window API function GlobalMemoryStatusEx which retrieves information about the system's current usage of both physical and virtual memory 32. For a Linux platform, the MMUA can invoke the Gestalt, sysconf and getsysinfo functions.

3.3. Agents migration

Each of these three mobile agents, described in Section 3.2, is a multi-task mobile agent. It can go through the servers to get information about the servers, and take the information back to its home agency. As shown in Fig. 6, after the monitoring server creates the agent, the agent will migrate to each of the host servers in sequence and return to its home agency with server usage information of all of the visited servers. When the agent returns to the home agency, we can retrieve the information that the agent takes back and store it in the database for further analyzing.

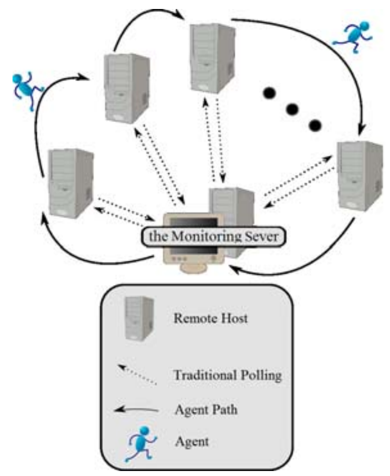
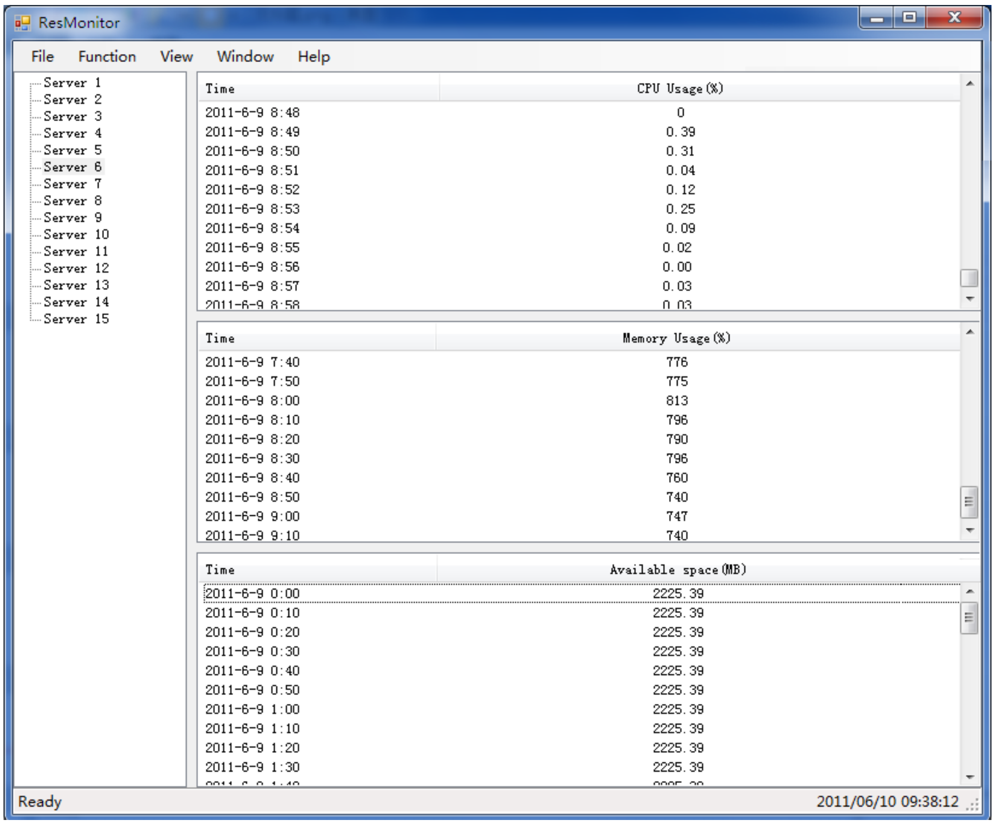


Fig. 6. The mobile agent technology VS traditional polling technology to get the performance information of the servers

The traditional polling method can only poll the servers one by one, as shown in Fig. 6. Comparing it with this information collecting using the traditional polling method, the mobile agent technology saves lots of the bandwidth of the network because the agents can travel through the servers. Atul Mishra and A.K. Sharma have proved that the mobile agents reduce bandwidth overloading problems by transferring the processing of the management data and decision making from the centralized management stations to the managed devices, thereby saving many repetitive request/response round trips 5.

3.4. Experiment on all servers in a server machine room

The experiment has been performed on all the servers in the server machine room. The monitoring server program, called the ResMonitor, is installed and run on one of PC server machines. Its interface is shown in Fig. 7. The client node program, called the ResReporter, is installed on each server in the server machine room.



Time	CPU Usage(%)
2011-6-9 8:48	0
2011-6-9 8:49	0.39
2011-6-9 8:50	0.31
2011-6-9 8:51	0.04
2011-6-9 8:52	0.12
2011-6-9 8:53	0.25
2011-6-9 8:54	0.09
2011-6-9 8:55	0.02
2011-6-9 8:56	0.00
2011-6-9 8:57	0.03
2011-6-9 8:58	0.03

Time	Memory Usage(%)
2011-6-9 7:40	776
2011-6-9 7:50	775
2011-6-9 8:00	813
2011-6-9 8:10	796
2011-6-9 8:20	790
2011-6-9 8:30	796
2011-6-9 8:40	760
2011-6-9 8:50	740
2011-6-9 9:00	747
2011-6-9 9:10	740

Time	Available space(MB)
2011-6-9 0:00	2225.39
2011-6-9 0:10	2225.39
2011-6-9 0:20	2225.39
2011-6-9 0:30	2225.39
2011-6-9 0:40	2225.39
2011-6-9 0:50	2225.39
2011-6-9 1:00	2225.39
2011-6-9 1:10	2225.39
2011-6-9 1:20	2225.39
2011-6-9 1:30	2225.39

Fig. 7. The interface of the monitoring server program

The MMUA and CPUUA are created by the monitoring server every 30 seconds, and the HDUA is created every 10 minutes. Each of them has 15 data collection tasks corresponding to the 15 servers. The experiment was conducted for 9 hours (8:00 AM to 5:00 PM). By analyzing the experiment data, the following problems, which we have never found before, have been identified.

(1) The hard drive C on a server used for ACM program contest has only 60 MB available space.

(2) The CPU usage of the student FTP server, which is used for collecting students' homework, is more than 70 % in daytime, and often reaches 100%.

3.5. Experiment on three main servers

The experiment was performed on three main servers in a server machine room. The network topology of the monitoring server and the three monitored servers are shown in Fig. 8. The monitoring server and the monitored servers are connected by a 1000 Mbps Ethernet. The MMUA and CPUUA are created by the monitoring server every 60 seconds, and the HDUA is created every 10 minutes. Each of them has three data collection tasks corresponding to the three servers. The experiment was conducted for 24 hours.

The distributions of CPU usage percentages of the three servers over 24 hours are plotted in Fig. 9. Three servers' CPU maximum usages were 39, 61, and 6%, respectively. In the students' frequently used time, which is from 7:30 AM to 8:30 PM, the CPU usage percentages of server I, server II and server III, are below 5, 40, and 3%, respectively, and in the rest time, the CPUs of the three servers were almost idle. Fig. 9 shows that the CPU load of the three main servers is light.

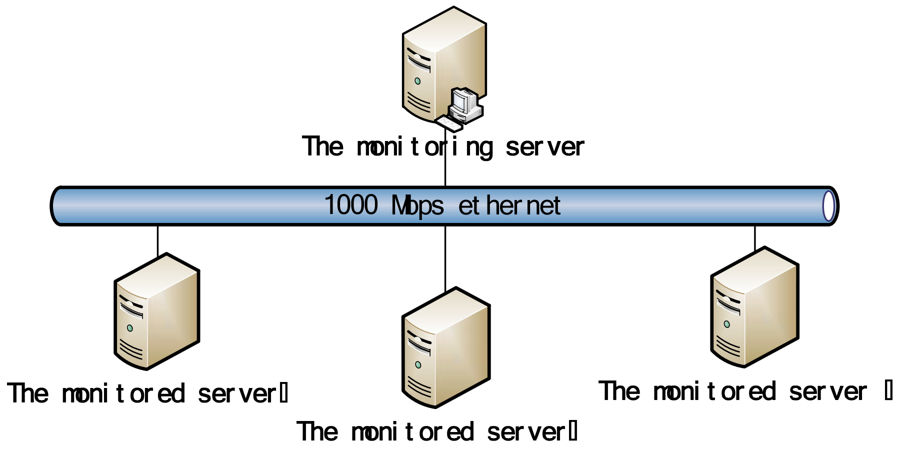


Fig. 8. The network topology of the servers in the experiment

The distributions of the memory usage percentages of the three servers over the 24 hours are shown in Fig. 10. The maximum memory usages of the three servers are 65.8, 49.2, and 39.5%, respectively. As shown in Fig. 10, the memory usage percentages of server I show an upward trend. During night time (0:00 AM to 7:30 AM, 21:30 PM to 24:00 PM), the memory usage percentages of server II and server III are low, while during day time (7:30AM to 21:30PM), they are high and fluctuating. This occurs because the computer center opening time is from 7:30 AM to 21:00 PM each day.

The distributions of the available spaces of the hard disk for the three servers over 24 hours, are plotted in Fig. 11. The minimum available spaces of the hard disk of the three servers were 2220, 15881, and 12683 MB, respectively. Although the disk available spaces of the three servers undulated in 24 hours, they tended to decline. The disk available spaces of server I were 2225.39 MB at 0:00 AM, down to 2221.67 MB at 23:50 PM. The disk available spaces of server II were 15955.91 MB at 0:00 AM, down to 15881.89 MB at 23:50 PM. The disk available spaces of server III were 12684.69 MB at 0:00 AM, down to 12683.24 MB at 23:50 PM. Based on these facts, the disk available spaces of the servers must always be checked.

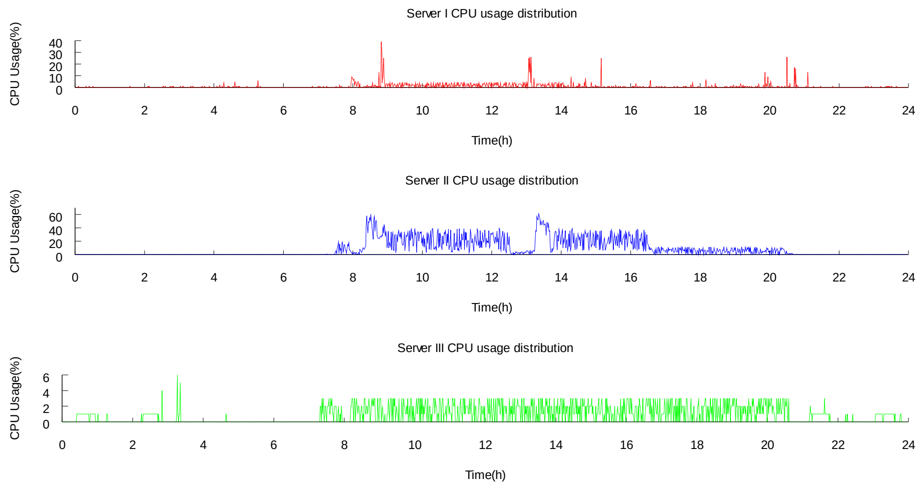


Fig. 9. The CPU usage distribution of the three monitored servers

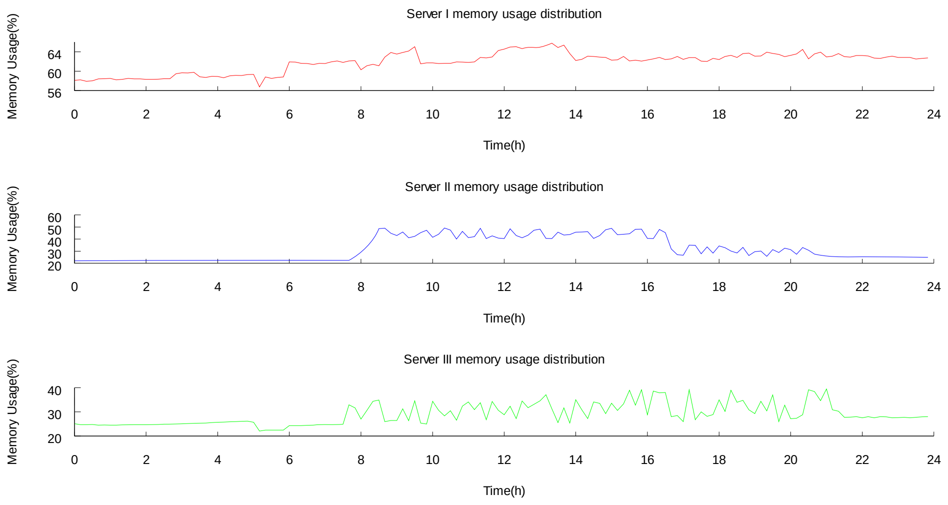


Fig. 10. The memory usage distribution of the three monitored servers

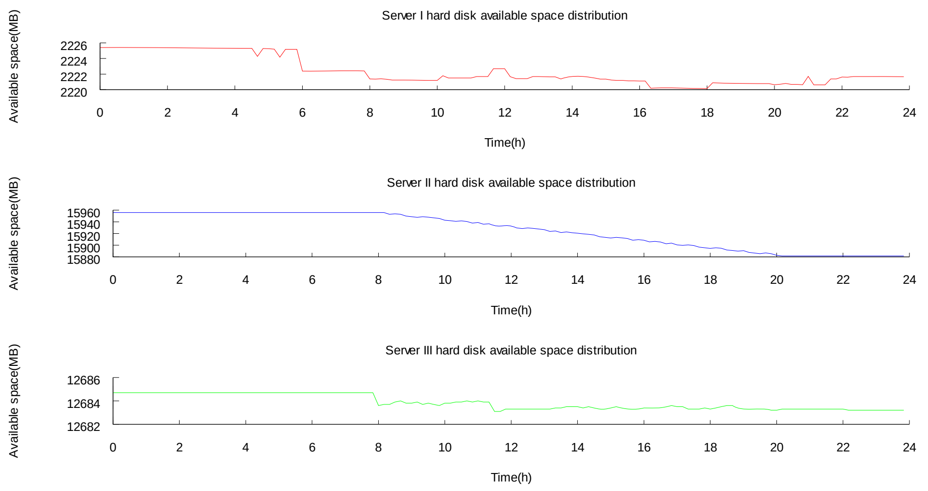


Fig. 11. The distribution of the available spaces of the hard disk of the three monitored servers

Based on the experimental data and the analysis above given, it can be concluded that the three main servers are in a good condition.

The client node program ResReporter has a small footprint. Because during the experiment, all services running the servers were going normally and smoothly, and in not very busy times, not only the CPU usage, but also the memory usage is kept at a relatively low using level.

4. Conclusions

A mobile agent-based system called Mobile Agent Based Computer Monitoring System (MABCMS) for monitoring the computer resource usage at public computer centers is presented. An IEEE FIPA compliant mobile agent system, called Mobile-C, is used as the base for MABCMS. The monitoring server can deploy resource monitoring algorithms to a group of or to all of the monitored client nodes easily, quickly, dynamically and silently. The experiment in a university computer center, with hundreds of computer workstations and 15 PC servers have been conducted to validate this system. The experiment uses MABSRMS to detect the system resources, such as available hard disk space, CPU usage and main memory usage. The experiment shows that the mobile agent-based monitoring system is a practical way to monitor the server resources in large scale distributed computer centers.

Acknowledgments: This work is supported by the Qianjiang Talent Project of Zhejiang Province under Grant No 2012R10056 and the Zhejiang Provincial Natural Science Foundation of China under Grant No LY13F020043.

R e f e r e n c e s

1. B a r t h, J. S. System and Method for Collecting and Displaying Information about Many Computer Systems. U. S. Patent No US 8131842B1, March 6, 2012.
2. S t a l l i n g s, W. SNMP, SNMPv2, SNMPv3 and RMON 1 and 2. Third Ed. Addison Wesley, 1999.
3. Z e n g, W., Y. W a n g. Design and Implementation of Server Monitoring System Based on SNMP. – In: Proceedings of 1st IITA International Joint Conference on Artificial Intelligence, China, Hainan Island, April 2009, 680-682.
4. Y u c h e n g, L., L. Y u b i n. A Monitoring System Design Program Based on B/S Mode. – In: 2010 International Conference on Intelligent Computation Technology and Automation, China, Changsha, May 2010, 184-187.
5. M i s h r a, A t u l, A. K. S h a r m a. Application of Mobile Agent in Distributed Network Management. – In: Proceedings of 2012 International Conference on Communication Systems and Network Technologies, India, Rajkot, Gujrat, May 2012, 930-935.
6. B i e s z c z a d, A., B. P a g u r e k, T. W h i t e. Mobile Agents for Network Management. – IEEE Communications Surveys, Vol. **1**, 1998, No 1, 1-9.
7. B e l l a v i s t a, P., A. C o r r a d i, C. S t e f a n e l l i. An Open Secure Mobile Agent Framework for Systems Management. – Journal of Network and Systems Management, Vol. **7**, 1999, No 3, 323-339.
8. A h n, J. Fault-Tolerant Mobile Agent-Based Monitoring Mechanism for Highly Dynamic Distributed Networks. – IJCSI International Journal of Computer Science Issues, Vol. **7**, May 2010, No 3, 1-7.
9. G a v a l a s, D., G. E. T s e k o u r a s, C. A n a g n o s t o p o u l o s. A Mobile Agent Platform for Distributed Network and System Management. – Journal of Systems and Software, Vol. **82**, 2009, 355-371.
10. N a i r, M. K., V. G o p a l a k r i s h n a. Applying Web Services with Mobile Agents For Computer Network Management. – International Journal of Computer Networks and Communications (IJCNC), Vol. **3**, March 2011, No 2, 125-144.
11. Z h a n g, N., J. Z h a n g. A Self-Adapted Anycast Routing Algorithm Based on Mobile Agent in Wireless Sensor Network. – Journal of Networks, Vol. **6**, 2011, No 2, 206-213.
12. W u, Q., N. S. V. R a o, J. B a r h e n et al. On Computing Mobile Agent Routes for Data Fusion in Distributed Sensor Networks. – IEEE Transactions on Knowledge and Data Engineering, Vol. **16**, June 2004, No 6.
13. W a n g, L. G., Y. D. Q i. Management Model Research of Low-Power Wireless Sensor Network. – Journal of Networks, Vol. **6**, 2011, No 12, 1734-1739.
14. U g u r, A., U. A. S i m a. Distributed Detection of Ddos Attacks During the Intermediate Phase Through Mobile Agents. – Computing and Informatics, Vol. **31**, 2012, No 4, 759-778.
15. E l K a d h i, N., K. H a d j a r, N. E l Z a n t. A Mobile Agents and Artificial Neural Networks for Intrusion Detection. – Journal of Software, Vol. **7**, 2012, No 1, 156-160.
16. H o u, Z., Z. Y u, W. Z h e n g, X. Z u o. Research on Distributed Intrusion Detection System Based on Mobile Agent. – Journal of Computers, Vol. **7**, 2012, No 8, 1919-1926.
17. Mobile-C: A Multi-Agent Platform for Mobile C/C++ Code. 2005. **http://www.mobilec.org**
18. C h e n, B o., H. H. C h e n g, J. P a l e n. Mobile-C: A Mobile Agent Platform for Mobile C/C++ Code. – Software – Practice & Experience, Vol. **36**, 2006, No 15, 1711-1733.
19. C h o u, Y.-C., D. K o, H. H. C h e n g. An Embeddable Mobile Agent Platform Supporting Runtime Code Mobility, Interaction and Coordination of Mobile Agents and Host Systems. – Information and Software Technology, Vol. **52**, February 2010, No 2, 185-196.
20. C h e n g, H. H. Scientific Computing in the Ch Programming Language. – Scientific Programming, Vol. **2**, 1993, No 3, 49-75.
21. C h e n g, H. H. Ch: A C/C++ Interpreter for Script Computing. – C/C++ User's Journal, Vol. **24**, 2006, No 1, 6-12.
22. C h e n g, H. H. Ch – An Embeddable C/C++ Interpreter. **http://www.softintegration.com**
23. Ch – An embeddable C/C++ Interpreter. Softintegration, Inc. **htth://www.softintergration.com/**

24. Softintegration. The Ch Language Environment – SDK User's Guide. Softintegration, Inc. **http//www.softintegration.com.**
25. Embedded Ch User's Guide, Softintegration, Inc. **http://www.softintegration.com/products/sdk/embedded ch/**
26. C h e n, B., H. H. C h e n g. Interpretive Open GL for Computer Graphics. – Computers & Graphics, Vol. **29**, June 2005, No 3, 331-339.
27. W a n g, Z., H. H. C h e n g. Portable C/C++ Code for Portable XML Data. – IEEE Software, Vol. **23**, No 1.
28. Microsoft Corporation. GetDiskFreeSpaceEx Function. 2013. **http://msdn.microsoft.com/en-us/library/aa364937(VS.85).aspx**
29. Microsoft Corporation. GetSystemTimes Function. 2013. **http://msdn.microsoft.com/en-us/library/windows/desktop/ms724400(v=vs.85).aspx**
30. Ejor. Get CPU Usage with GetSystemTimes. December 2004. **http://www.codeproject.com/Articles/9113/Get-CPU-Usage-with-GetSystemTimes**
31. Sascha Nitsch Unternehmensberatung UG. www.LinuxHowtos.org howtos, tips&tricks and tutorials for linux. 2013. **http://www.linuxhowtos.org/System/procstat.htm**
32. Microsoft Corporation. GlobalMemoryStatusEx Function. 2013. **http://msdn.microsoft.com/en-us/library/windows/desktop/aa366589(v=vs.85).aspx**