Efficient Use of Mobile Agents in the Management of Distributed Systems

[1]H. M. Kelash, [2]M. Amoon, [3]H. M. Faheem

[12]Dept. of Computer Science & Eng., Faculty of Electronic Engineering, Menofia University, Menouf, 32952, Egypt, m_amoon74@yahoo.com

[3]Dept. of Computer Science,Faculty of Information & Computers, Ain Shams University, Cairo, Egypt hmfaheem@btexperts.com

Abstract

Distributed systems management has steadily evolved from a centralized paradigm, where all the management processing takes place in a single management station, to distributed paradigms, where management is distributed over a potentially large number of nodes. Mobile agent technology conserves network bandwidth and improves management efficiency by decreasing network traffic. Furthermore, the use of mobile agent technology to distribute and delegate tasks promises to overcome scalability and flexibility limitations of the centralized management paradigms. This paper proposes a new multiagent based system for management of distributed systems. This proposed system adopts mobile agents as a technology for achieving management tasks. The presented results are based on a design example of an application operating in a mobile environment.

I. INTRODUCTION

The increasing development of distributed systems requires more efficient technologies to ensure the access and the management of the network resources. Features such as efficiency, flexibility, velocity, and scalability should be related to all the operations performed within distributed systems [1]. For growing Internet, it is very important to have up-to-date information about your systems components and it becomes more important if you are Internet provider. Increasing size and complexity of network and the need to support all sorts of advanced services in a reliable and cost-effective manner pose a major challenge to network management [2].

The primary goal of distributed systems management is to ensure efficient use of resources and provide timely service to users. Management has evolved from simple centralized static solutions to dynamic web-based solutions involving heterogeneous computing systems [3].

Current Management systems are typically designed according to a centralized Network Management (NM) paradigm characterized by a low degree of flexibility and re-configurability. Management interactions are based on a centralized, client/server model, where a central station (manager) collects, aggregates and processes data retrieved from physically distributed servers. Widely deployed NM standards, such as the Simple Network Management Protocol (SNMP) [4] of the TCP/IP protocol suite are designed according to this rigid centralized model.

Within these protocols, physical resources are represented by managed objects. Collections of managed objects are grouped into tree-structured Management Information Bases (MIB2).

The idea of management distribution is taken further by solutions that exploit Mobile Agents (MA), which provide a powerful software interaction paradigm that allows code migration between hosts for remote execution [5]. The data throughput problem can be addressed by delegation of authority from managers to MAs, which are able to filter and process data locally without the need for transmission to a central manager [6]. This ability has attracted much attention to MA technology, with several Mobile Agent Frameworks (MAF) proposed for NM applications [7]-[10].

In this paper, a multiagent system for the management of resources in distributed systems is presented and studied. The system depends on the use of Mobile Agent Technology (MAT) to achieve distributed management.

The rest of this paper is organized as follows: In section II, evolution of distributed systems management is presented. Section III discusses the main categories of the management operations. Section IV describes the structure and the operation of the proposed multiagent system. In section V, the performance and results are discussed. Finally, we present conclusions in section VI.

II. MANAGEMENT EVOLUTION

Distributed systems began to appear into production during the early 1970's, with the advent of mini computer. These systems were often isolated and used for departmental and other specialized functions. After that, Personal Computer (PC) and Local Area Network (LAN) technology were appeared. This technology provided new possibilities and challenges. Software and hardware resources could be shared and easily accessed. Shared resources required performance analysis and tuning to reduce systems response time and improve productivity.

Then, the client server applications appeared with distribution some of the computational requirements to the PC, moving from the previously centralized processing. This paradigm added entirely new areas of management. Synchronized application upgrades were required along with tools to monitor and troubleshoot performance and other network problems. Therefore, new industry standards had emerged to provide a common mechanism to manage the network. The SNMP became the industry standard for network management. Vendors of networked systems installed software agents within their hardware to enable remote access to their components. This protocol very quickly became the accepted standard for networked hardware management [11].

By the late 1990's brought yet another factor to the distributed management with the appearance of Mobile Agent Technology (MAT). Mobile agent technology has been considered an enhancement of distributed technologies as it provides powerful and efficient mechanisms to develop applications for distributed and heterogeneous systems. There is an increasing amount of data/resources available nowadays in distributed systems and a large number of tasks that have to be performed to manipulate these data/resources. Mobile agent technology offers the possibility of executing these tasks in an automated way, with minimal human intervention [12-14].

Management tasks are assigned to a mobile agent and that agent can be sent to remote hosts to execute the assigned tasks. After executing the assignment, the results are carried back to the host (sender) by the same agent. Thus, the conventional centralized management is converted into a distributed management environment [15].

III. MANAGEMENT OPERATIONS

Two main categories of management operations have been identified among the user requirements: the *read* and the *set* management operations. The first category includes the monitoring of the resource. The second category refers to the set or modification of resource parameters [1]. Now, we will summarize some of management operations that can be applied on resources in distributed systems. The operations belong to the above two categories and include:

- *Monitoring the value of parameter X belonging to the resource R every N seconds. Do operation Z and notify the source node with the result.*

This operation implies reading the parameter value in a local MIB, which is associated to the resource *R*, and the validation of this value. A client-server solution creates a separate thread within the NM application that every *N* seconds sends an interrogation message to the local MIB. The NMS verifies the returned parameter value and decides if it is valid or not. In the case of a fault, it sends a message to the local MIB to execute operation *Z*, and a notification message to the user.

A mobile agent solution sends an agent designed to perform this query to the subnetwork where the resource is connected. The agent reads and validates the parameter value every *N* seconds, using local interaction. In the case of a fault, the agent executes operation *Z* and then sends a notification message to the system manager that will forward it to the user.

The client-server solution uses remote interaction, which, in this case, has two important disadvantages: it increases the traffic in the network and introduces an overhead in the execution of the query. The mobile agent approach eliminates these two disadvantages by using local interaction. Further, in the case of a fault, the mobile agent immediately executes operation *Z* at the device side. Time may play an important role in a high-traffic network within real-time systems.

- *A new resource R has been added to a subnetwork in the distributed system. Update the related information in the MIB Database.*

When adding a new resource in the network, the MIB database should be updated. In the client-server approach, there is a pre-defined protocol established between the newly added device and MIB database. There are cases when the protocol must be changed or personalized for a specific resource due to the new or changed features of the resource. In client-server architecture, such a protocol would need complex modifications that involve upgrading of the software in the whole system and this may be costly or inefficient.

A mobile agent based solution sends an agent to access the MIB Database to add the information related to presence of the new resource in the system. The mobile logic of the agent can be changed or personalized for each new added resource, without modifying any of the other applications in the system. Mobile logic gives an important degree of flexibility to the design of the distributed systems. Further, the scalability of the system is implicitly ensured.

This operation can be also used in the case in which a device becomes unavailable due to a temporary or a permanent fault. The direct advantage is that any fault can be immediately identified and fixed; an aspect that is very important in applicable industrial systems.

- *List all the type T resources of the distributed system.*

This type of operation implies searching every subnetwork of the distributed system that has at least one resource of type *T* connected to it. This operation has to be sent to each MIB database in all subnetworks in order to search for the type *T* resources, and to check its status. It is clear that, this operation generates an enormous traffic that under several circumstances causes the network to be congested. In a client-server approach, the NMS creates a number of threads equal to the number of subnetworks. After receiving all the partial results from each subnetwork, the NMS puts them together and sends the final result to the system management application.

An alternative client-server-based solution may create only one thread in the NMS that will sequentially interrogate all the subnetworks and then will provide the result to the user. The first solution performs the operation in a minimum time, but introduces a computing overhead in the system manager application. The NMS has to handle all the requests and answers sent and received to/from the subnetworks, and this means a high load for its resources and a decrease in its performances. The second solution generates only one thread within the system manager application, but it will need significantly more time to perform the query. Both solutions generate the same number of request-answer messages to execute the query.

A solution based on mobile agent technology implies the creation of an agent designed to migrate from subnetwork to subnetwork searching for the type *T* resources. The NMS is responsible for creating the mobile agent and for receiving the result of the query provided by the agent. Finally, it forwards the result to the user. The mobile agent based solution achieves load balancing in the whole system by the agent migration. It improves the time taken to perform the operation by using the local interaction. The traffic in the network is due only to the migration of the agent from host to host (in comparison to the 2*n* messages exchanged in a client-server architecture; where *n* is the number of subnetworks).

IV. DISTRIBUTED MANAGEMENT SYSTEM

The purpose of the proposed multiagent system is to transparently locate, monitor and manage resources in distributed systems. The system consists of a set of static and mobile agents. Some of them reside in each node or element in the distributed system. There are two mobile agents named delegated and collector agents that can move through the distributed system. The role of each agent in the multiagent system, the interaction between agents, and the operation of the system are described in the following subsection.

A. Multiagent System Structure

The multiagent system structure assumes that each node in the system will have a set of agents residing and running on that node. These agent types are:

- ***Client agent*** (CA) percepts service requests, initiated by the user, from the system. The request of the user is passed to client agent. The CA may receive the request from the local user directly. In the other case, it will receive the request from the exporter agent coming from another node. Each node has a single client agent. The user request may be asking for data or status information from a resource or may be sending data or management information to a resource. The resource is either local or remote.
- ***Service list agent*** (SLA) maintains a list of the resource agents that reside on each node. This agent will receive the request from the client agent and send it to the resource availability agent. If the reply indicates that the requested resource is local then the service list agent will deliver the user request to the categorizer agent. Otherwise, it will deliver the request to the exporter agent. Each node has a single SLA.
- ***Resource availability agent*** (RAA) indicates whether the requested resource is free and available for use or not. It also indicates whether the requested resource is local or remote. It receives the request from the service list agent and checks the status of the requested resource through the access of a database that contains status information of the system resources. The agent then constructs the reply (proper action) depending on the retrieved information from the database. Each node has a single resource availability agent.
- ***Resource agent*** (RSA), which is responsible for the operation and control of the resource. This agent executes the user service request on the resource. Each resource per the system has one or more resource agents. Each node may have zero or more RSAs.
- ***Router agent*** (RA) provides the path of the requested resource on the network in case of accessing remote resources. Before being dispatched, the exporter agent will ask the router agent for the path of the requested resource. This in turn delivers it to the exporter agent. Each node has a single RA.
- ***Categorizer agent*** (CZA) allocates a suitable resource agent to perform the user request. This agent percepts inputs coming from the service list agent. It then tries to find a suitable free resource agent to perform the requested service. Each node has a single categorizer agent.
- ***Exporter agent*** (EA) is a mobile agent that can carry the user request through the path identified by the router agent to reach the node that has the required resource. It passes the requested resource id to the router agent and then receives the reply. If the router agent has no information about the requested resource, the EA will try to locate the resource in the system. Each node may have several exporter agents.

There are also two additional mobile agent types exist in the system:

- ***Delegated agent*** (DA) is a mobile agent that is launched in each subnetwork. It is responsible for traversing subnetwork nodes instead of the exporter agent to do the required task and carry results back to the exporter agent; and
- ***Collector agent*** (CTA) is a mobile agent that is launched from the last subnetwork visited by the exporter agent. It is launched when results from that subnetwork become available. This agent goes through the reversed itinerary of the exporter agent trip. The CTA collects results from the delegated agents and carries it to the source node.

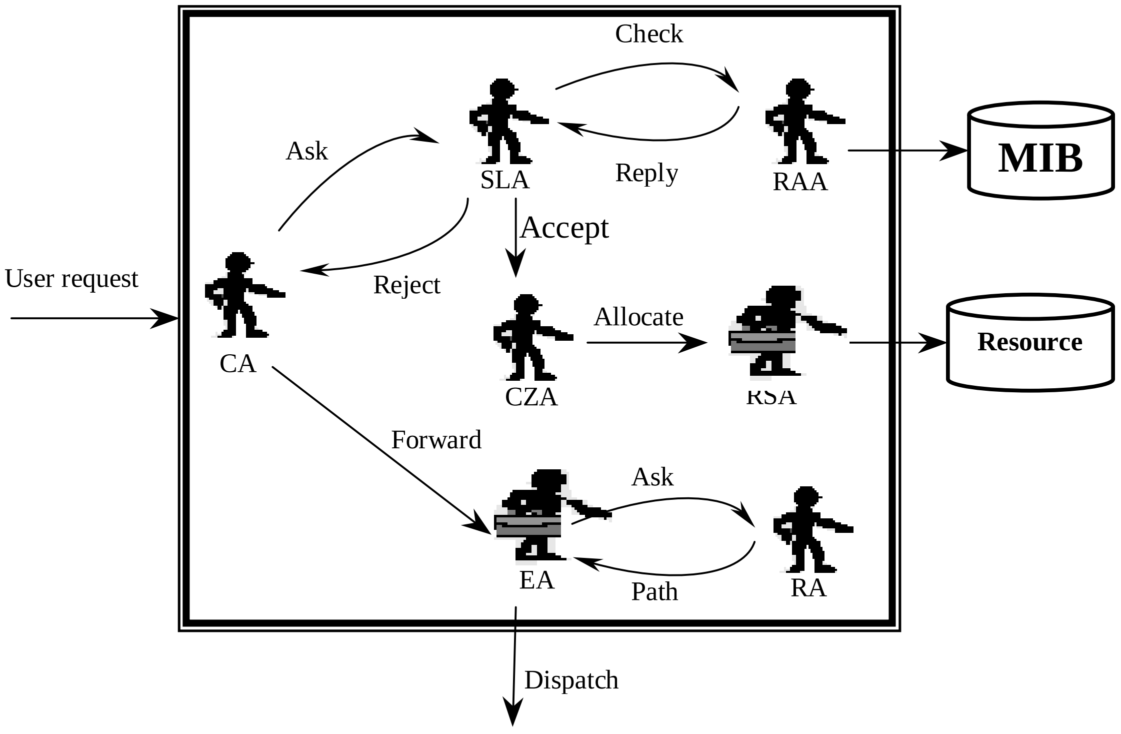

Fig. 1. Agent activity cycle.

B. System's Operation

The activity cycle of the proposed multiagent system is shown in Fig. 1. The client agent receives the service requests either from the user or from an exporter agent dispatched from another NE. The client agent issues a request to the service list agent for a service resource agent. The service list agent consults a resource availability agent to advise whether a resource is available locally or remotely. If the resource is locally available then the service list agent will then forward a request to the categorizer agent to allocate a suitable resource agent to start performing the required task on a specified resource. Otherwise, the service list agent informs the client agent with the rejection. The client agent forwards the request to the exporter agent.

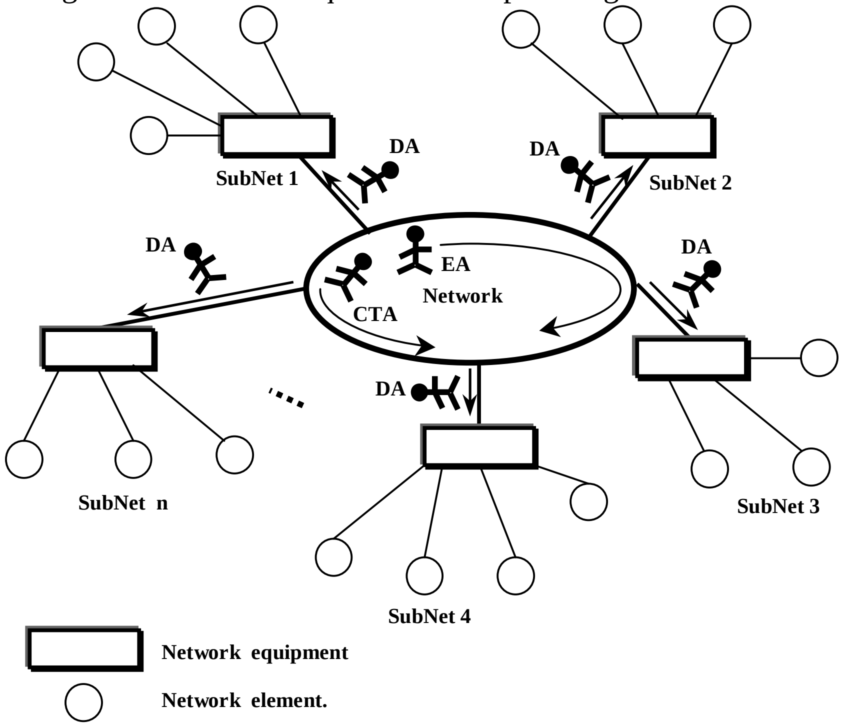

Fig. 2. The network architecture of the distributed system.

The exporter agent forwards a request to the router agent to assign a path. The exporter agent will be dispatched through the network channel to the destination node identified by that path. If the request coming from an exporter agent dispatched from another NE the client agent will inform that exporter agent with the rejection. This exporter agent then resumes its trip through the network.

As shown in Fig. 2, the exporter agent traverses the subnetworks of the distributed system through its trip. At each subnetwork a delegated agent is launched to traverse the local nodes of that subnetwork doing the required task and carrying results of that task There are two approaches to collect results of the required task and send it back to the source: -

- The exporter agent can wait at each visited subnetwork until the result is obtained and carry it to the next subnetwork in its itinerary. The wait of the exporter agent prevents the tasks execution to be started in the other subnetworks. This approach is the used approach in most of the previously developed management systems that their operation is based on mobile agents [2], [7], [10].
- The exporter agent does not wait the results of that subnetwork. It resumes its trip visiting other subnetworks and at each subnetwork another delegated mobile agent is launched. The exporter agent will be killed at the last subnetwork visited in its itinerary. When the results from that subnetwork become available, another mobile agent called collector agent is launched from this subnetwork to collect results from the other subnetworks. The collector agent goes through the reversed itinerary of the exporter agent trip. In this manner, operations can be done in a parallel fashion at subnetworks because there is no delay of the task submission to nodes in these subnetworks. This approach is the used one for collecting results in the proposed system.

V. PERFORMANCE ANALYSIS AND RESULTS

A. Performance Analysis

The performance of the proposed system is measured in terms of the response time, which is the time between dispatching tasks from the source node and the returning of results. The main concern of performance improvement of network applications or services is to reduce the overall response time [7], [16].

The proposed system is characterized by the following variables:

n	number of subnetworks in the distributed system;
t_b	average link delay time between two subnetworks;
t_d	time for interaction between the exporter agent and the delegated agent;
t_e	average time to complete a task in a subnetwork.

For the client server case, the source node will need to construct a request message as many n times. Thus the total response time in the average case is the sum

$$T_0 = 2nt_b + nt_e. \qquad (1)$$

In traditional management systems that uses mobile agents, the EA must wait at each subnetwork to obtain the result and thus the total response time in the average case is the sum

$$T_1 = nt_b + 2\,(n - 1)\,t_d + (n - 1)\,t_e. \qquad (2)$$

In our proposed system, the response time is the sum of exporter agent time and the collector agent time. The exporter agent time is the sum

$$T_{EA} = (n - 1)\,t_b + (n - 1)\,t_d. \qquad (3)$$

The collector agent time is sum

$$T_{CTA} = (n - 1)\, t_b + (n - 1)\, t_d + t_e. \tag{4}$$

The last t_e is the task execution time at the last received subnetwork. Thus, the total response time of our proposed system is

$$T_2 = T_{EA} + T_{CTA} = 2(n - 1)\, t_b + 2(n - 1)\, t_d + t_e. \tag{5}$$

The speedup obtained by the proposed system over the traditional mobile agent-based management systems is calculated as

$$S = \frac{T_1}{T_2}. \tag{6}$$

The efficiency of distributed systems is calculated as

$$E = \frac{Speedup}{n}. \tag{7}$$

B. Results

The proposed multiagent system is implemented using IBM Aglets [17] with java virtual machine (JVM) as the platform and the development package is the Aglets Software Development Kit (ASDK). The proposed system is applied on a campus-distributed system that is partitioned into 32 subnetworks and each subnetwork contains from 5 to 10 nodes. The proposed multiagent system is implemented at each node of the system and is run under the Aglets platform.

As a case study, we will address the use of the proposed multiagent system for the management of a database element as a software resource. The implemented multiagent system will be used to locate this element in the distributed system by searching the whole system. It can then monitor the status of that element through accessing the MIB and send/receive data from and to it through a simple query. The response time of locating the resource, sending tasks to the resource, and receiving results from a database resource residing in a computer node in the distributed system using the proposed system is measured. This time is measured for different link speeds such as 100Mbps, 1000Mbps, and 10,000Mbps. The response time is then compared against both centralized management systems and traditional mobile agent management systems. This comparison is deployed in Fig. 3. Both speedup and efficiency are also calculated and reported in Fig. 4.

Fig. 3 shows that, the proposed multiagent system provides response time better than that obtained by the centralized approach and the traditional mobile management systems for different link transfer rates. It is also clear that as the number of subnetworks in the system increases the response time increases. The rate of this increment in the proposed system is lower than the rate in case of the other two traditional management models.

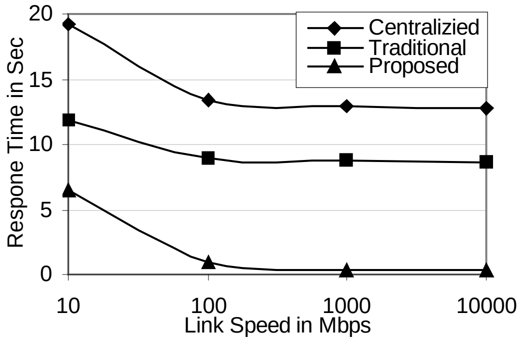

Fig.3. Response time comparison.

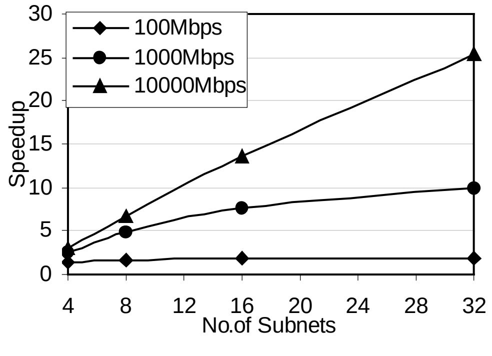

Fig.4. Speedup comparison.

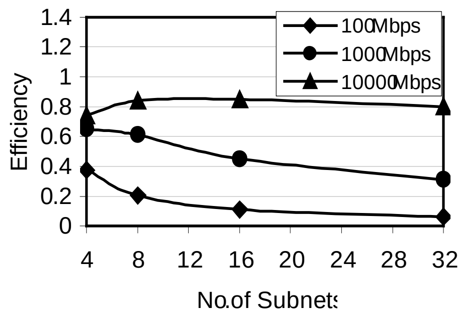

Fig.5. Efficiency comparison.

Fig. 3 shows that, the proposed multiagent system provides response time better than that obtained by the centralized approach and the traditional mobile management systems for different link transfer rates. It is also clear that as the number of subnetworks in the system increases the response time increases. The rate of this increment in the proposed system is lower than the rate in case of the other two traditional management models.

Fig. 4 shows that there is a clear speedup obtained from using the proposed system over using the traditional mobile management systems. It is shown that speedup is better for high transfer data rates. The low rate increment in the response time and the high-obtained speedup means that the proposed system reflects better scalability than the other traditional management systems.

Fig. 5 shows that the proposed system has an efficient performance compared with traditional mobile management systems. Another advantage of the proposed system is portability because the JVM is used as the development platform and thus it can run under any operating system platform.

VI. CONCLUSION

Distributed management for distributed systems is becoming a reality due to the rapid growing trend in internetworking and the rapid expanding connectivity. This work describes a multiagent system for the management of distributed systems. The proposed system can locates, monitors and manages resources in the system. The new technique in that system allows that the management tasks to be submitted to subnetworks of the distributed system and executed in a parallel fashion. The proposed system uses two mobile agents. The first is used to submit tasks to nodes and the other collects the results from these nodes. The proposed system is compared against the traditional management technique in terms of response time, speedup, and efficiency. A prototype has been implemented using the performance management as the case study. The performance results indicate a significant improvement in response time, speedup, efficiency, scalability, and portability than the traditional techniques.

REFERENCES:

1. C. Raibulet and C. Demartini, “Mobile Agent Technology for the Management of Distributed Systems – a Case Study,” *in the proc. of TERENA Networking conf.*, Portugal, May 2000.
2. U. Shanker, “Autonomous and Mobile Agents in Distributed Network Management and Monitoring System,” *in Workshop on Distributed Computing on the WEB*, Germany, June 22-23, 1998.
3. H. Abdu, H. Lutfiyya, and M. A. Bauer, “A Model for Efficient Configuration of Management Agents in Distributed Systems,” *in ACM Performance Evaluation Journal*, Vol. 54, Issue 4, pp. 285-309, Dec. 2003.
4. C. Pattinson, “A Study of the Behaviour of the Simple Network Management Protocol,” *in the 12th International Workshop on Distributed Systems: Operations and Management DSOM'2001*, Nancy France, October 15-17, 2001.
5. J. Waldo, “Mobile Code, Distributed Computing, and Agents,” *in the proc. of IEE Intelligent Systems*, March/April 2001.
6. D. Gavalas, D. Greenwood, M. Ghanbari, M. O’Mahony, “Advanced Network Monitoring Applications Based on Mobile/Intelligent Agent Technology,” *in the proc. of IEEE International Conference on*, Vol. 2, pp. 1362-1366, June 6-10, 1999.
7. H. Ku, G. W. R. Ludere, and B. Subbiah, “An Intelligent Mobile Agent Framework for Distributed Network Management,” *in Globecom’97 Phoenix* conf., AZ, Nov 1997.
8. M. G. Rubinstein, Otto C. M. B. Duarte, and G. Pujolle, “Using Mobile Agent Strategies for Reducing the Response Time in Network Management,” *in the proc. of the 16th IFIP International conf. World Computer Congress*, pp. 278-281, China, August 2000.
9. D. Raz and Y. Shavitt, “Toward Efficient Distributed Network Management,” *in Journal of Network and Systems Management*, Vol. 9, No. 3, September 2001.
10. T. C. Du, E. Y. L1, and A. Chang, “Mobile Agents in Distributed Network Management,” *in Communication of the ACM*, Vol. 46, No. 7, pp.127-132, July 2003.
11. A. Westerinen and W. Bumpus, “The Continuing Evolution of Distributed Systems Management,” *in IEICE trans. of INF. & SYST.*, Vol. E86-D, No. 11, November 2003.
12. J. Waldo, “Mobile Code, Distributed Computing, and Agents,” *in the proceedings of IEE Intelligent Systems*, March/April 2001.
13. [13] D. G. A. Mobach, B. J. Overeinder, N. J. E. wijngaards, and F. M. T. Brazier, “Managing Agent Life Cycles in Open Distributed Systems,” *in the proc. of the 18th ACM Symposium on Applied Computing*, pp. 61-65, USA 2003.
14. [14] S. Kleijkers, F. Wisman, and N. Roos, “A Mobile Multi-Agent System for Distributed Computing,” *in lecture notes in computer science*, Vol. 2530/2003, pp. 158-163, august 2003.
15. C. Huang and C. Pattinson, “Using Mobile Agent Techniques for Distributed Manufacturing Network Management,” *in the proc. of PGNet 2nd Annual conf.*, Liverpool, 2001.
16. M. G. Rubinstein, O. Carlos, M. B. Duarte, and G. Pujolle, “Improving Management Performance by Using Multiple Mobile,” *in the pro. of the 4th International conf. on Autonomous Agents-ACM Agents 2000*, pp. 165-166, Spain, June 2000.
17. D. B. Lange, M. Oshima, “Mobile Agents with Java: The Aglet API1,” *http://turtle.ee.ncku.edu.tw/~ac/homepage/project/Aglet.*

Article received: 2006-02-25