A Multiagent System for Distributed Systems Management

H. M. Kelash, H. M. Faheem, and M. Amoon

***Abstract*—**The demand for autonomous resource management for distributed systems has increased in recent years. Distributed systems require an efficient and powerful communication mechanism between applications running on different hosts and networks. The use of mobile agent technology to distribute and delegate management tasks promises to overcome the scalability and flexibility limitations of the currently used centralized management approach. This work proposes a multiagent system that adopts mobile agents as a technology for tasks distribution, results collection, and management of resources in large-scale distributed systems. A new mobile agent-based approach for collecting results from distributed system elements is presented. The technique of artificial intelligence based on intelligent agents giving the system a proactive behavior. The presented results are based on a design example of an application operating in a mobile environment.

***Keywords*—** distributed management, distributed systems, efficiency, mobile agent, multiagent, response time

I. INTRODUCTION

BUILDING and managing large-scale distributed systems is becoming an increasingly challenging task. Continuous intervention by user administrators is generally limited in large-scale distributed environments. System support is also needed for configuration and reorganization when systems evolve with the addition of new resources [1]. The increasing development of distributed systems requires more efficient technologies to ensure efficient management for system resources. Distributed system management applications are usually based in one of two classic protocols: Simple Network Management Protocol (SNMP) widely deployed in IP networks, and Common Management Information Protocol (CMIP) for telecommunication networks that are both centralized approaches [2-5].

Centralized Network Management Systems (NMSs) are clients to management agents residing permanently in each managed network element (NE). Although adequate for most practical management applications, the limitations of SNMP-for example, the potential processing and traffic bottleneck at the NMS-have been recognized for many years [2], [6].

Manuscript received January 9, 2006.

K. M. Kelash is with the Computer Science & Eng. Department, Menofia University, CO 32952 EGYPT (048-366-0716; fax: 048-366-0716; e-mail: mohammed_amoon@hotmail.com).

H. M. Faheem is with the Dept. of Computer Science, Ain Shamas University, Cairo, EGYPT (e-mail: hmfaheem@btexperts.com).

M. Amoon is with the Computer Science & Eng. Department, Menofia University, CO 32952 EGYPT (048-366-0716; fax: 048-366-0716; e-mail:

It has been observed that distributed system management functions may achieve several benefits [2–4]. This type of management offers several perceived advantages such as network traffic and processing load in the NMS can be both reduced by performing data processing closer to the NEs; scalability to large networks is improved; searches can be performed closer to the data, improving speed and efficiency; and distributed system management is inherently more robust without depending on continuous communications between the NMS and NEs. Recent attention to distributed system management has increased because the processing capabilities of routers and switches have improved considerably and the popularization of Java and CORBA has brought mobile code concepts closer to mainstream acceptance [5], [7].

The need for more scalable and flexible Distributed system management applications were leading to an intensive quest for higher decentralization, with approaches like Management by Delegation, Web-based Management, Intelligent Agents, Active Networks and, more recently, Mobile Agent Technology (MAT) [8].

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 [9-12].

This paper attempts to specify a multiagent system for the management of resources in large-scale distributed systems. The system includes the function of each agent and the interactions that occur among agents.

The rest of this paper is structured as follows: Section II contains general considerations about distributed systems, the distributed system management approaches, and a brief description of mobile agents. 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 our conclusions in section VI.

II. DISTRIBUTED SYSTEMS AND MOBILITY

A. Distributed Systems

A distributed system is one in which components located at networked computers communicate and coordinate their actions only by passing messages [6]. The motivation for constructing and using distributed systems stems from a desire to share resources. These resources may be hardware components such as disks and printers or software-defined

entities such as files, databases, and data objects of all kinds [6]. The primary goal of the management of distributed systems 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 [13], [14].

Most of the distributed system management techniques still follow the centralized model that is based on the client-server model. Centralization presents some problems, such as: (1) it could cause a traffic overload and processing at the manager may affect its performance; (2) it does not present scalability in the increase of the complexity of the network; (3) the fault in the central manager node can leave the system without management [12], [15].

One model is the distributed management where management tasks are spread across the managed infrastructure and are carried out at managed resources. The goal is to minimize the network traffic related to management and to speedup management tasks by distributing operations across resources. The new trend in distributed system management involves using mobile agents to manage the resources of distributed systems [2], [9], [10], [12].

B. Management Approaches

A resource management approach can have three main functions: monitoring, analyzing, and controlling. The first function is to *monitor* the performance of applications and the resources that are currently being used by applications. This monitoring requires collecting information about resources from the Management Information Base (MIB) database. The MIB database contains many data objects for network management such as system data, resource status, and communication statues. The MIB is defined in MIB-II of RFC-1213 [16]. Secondly, the observed performance of an application is compared to the required performance of the application. When real-time requirements (deadlines) are violated, the resource management approach must *analyze* the situation to determine what applications, communication sub-paths, or resources are causing the violation. Finally, the approach must take a corrective action by *controlling* the allocation of applications to resources such that required real-time performance is again achieved [17].

All the distributed management techniques used in the literature can be classified into four approaches. These four management approaches are shown in Fig. 1. In Fig. 1(a), the *client-server* (*CS*) model represents the SNMP paradigm where a centralized NMS polls a network of network elements. The communication between the NMS and agents is characterized by pairs of query response messages for every interaction. Fig. 1(b) represents a *hierarchical static* (*HS*) approach modeled as *L* midlevel managers, each managing a separate subnetwork of *N*/*L* network elements. A two-level hierarchy is considered here although multiple hierarchical layers may evidently be possible. Each subnetwork is managed in a client-server manner and midlevel managers may communicate with a centralized high-level NMS as needed.

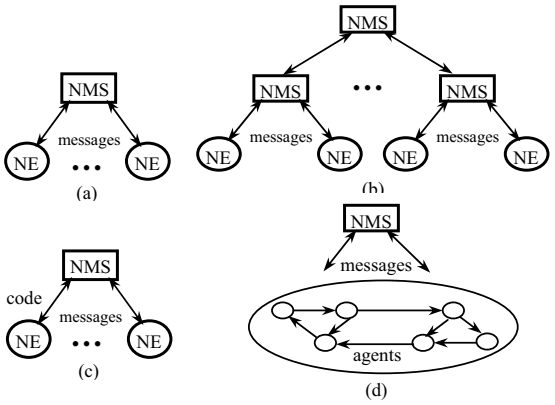


Fig. 1 Centralized and distributed network management approaches. (a) Client-server. (b) Hierarchical static. (c) Weak mobility. (d) Strong mobility.

In the *weak mobility* (*WM*) approach, the NMS distributes code to specific NEs where the code is executed, as shown in Fig. 1(c). After performing its specific task, the code will typically report results back to the NMS and expire at the NE. During execution, the code does not have a capability for autonomous migration to other NEs. The highest degree of mobility is the *strong mobility* (*SM*) [18]. In the *SM* approach, the NMS dispatches one or more agents to carry out a specific task, as shown in Fig. 1(d). Agents have the capability to autonomously travel (execution state and code) among different NEs to complete their task. The route may be predetermined or chosen dynamically depending on the results at each NE [2]. The proposed multiagent system uses the strong mobility approach.

C. Mobility

The concept of mobile agents promises new ways of designing applications that better use the resources and services of computer systems and networks. For example, moving a program (e.g. search engine) to a resource (e.g. database) can save a lot of bandwidth and can be an enabling factor for applications which otherwise would not be practical due to network latency.

Mobile agent is an execution unit, which is able to migrate autonomously to another host and resume execution seamlessly. Conceptually, a mobile agent can migrate its whole virtual machine from host to host: it owns the code, not the resources. Mobile agents are the basis of an emerging technology that promises to make it very much easier to design, implement, and maintain distributed systems. We have found that mobile agents reduce network traffic, provide an effective means of overcoming network latency, and perhaps most importantly, through their ability to operate asynchronously and autonomously of the process that created them, help us to construct more robust and fault tolerant systems [19].

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 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 Management Information Base (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.

- *A new resource R has been added to a subnetwork in the distributed system.*

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.

- *List all the type T resources in the 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.

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 $2n$ 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 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 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.
- ***Service list agent*** (SLA) has a list of the resource agents in the system. This agent will receive the request from the CA 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 request to the categorizer agent. Otherwise, it will return the request to the CA.
- ***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 the MIB. The agent then constructs the reply depending on the retrieved information from the database.
- ***Resource agent*** (RSA) is responsible for the operation and control of the resource. This agent executes the on the resource. 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.
- ***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.

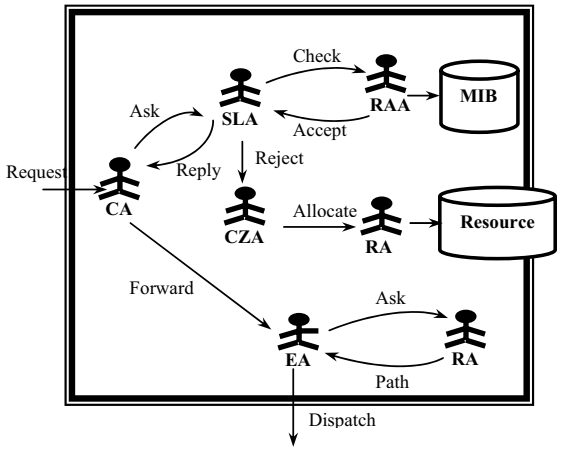


Fig. 2. Node activity cycle.

- ***Exporter agent*** (EA) is a mobile agent that can carry the user request through the path identified by the RA to reach the node that has the required resource. It passes the requested resource *id* to the RA 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.

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.
- ***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.

B. System's Operation

The activity cycle of our multiagent system residing in a node is shown in the Fig. 2. The client agent receives the service requests either from the user or from an exporter agent. The client agent then asks the service list agent for the existence of a resource agent that can perform the request. The service list agent checks the availability of the required resource agent, by consulting a resource availability agent, to perform the requested service. The reply of the resource availability agent describes whether the resource is locally available or not and whether there is a resource agent that can perform the requested service or not. In case of the resource availability agent accepts the request then, the service list agent will ask the categorizer agent to allocate a suitable resource agent to the requested service and then the resource agent performs the requested service. Otherwise, the service list agent informs the client agent with the rejection and is then passed to the exporter agent.

The exporter agent then asks the router agent for the path of the required resource agent. Once the path is determined, the exporter agent will be dispatched through the network channel to the destination node identified by that path. If the router agent has no information about the location of the required resource agent, then the exporter agent will search the distributed system to find the location of the required resource agent and assigns the required task to it.

As shown in Fig. 3, 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: -

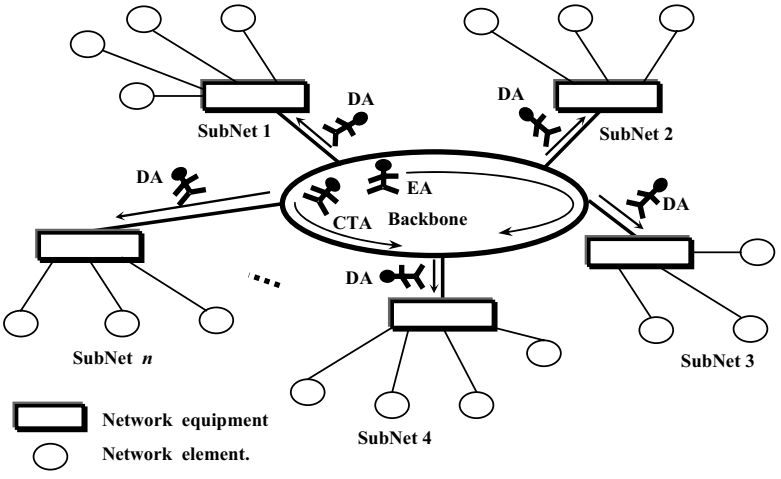


Fig. 3 The network architecture of the distributed system

- 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.
- The exporter agent does not wait the results from that subnetwork. It resumes its trip visiting other subnetworks and at each subnetwork another DA 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 is the used approach for collecting results in our 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 [4], [19].

The proposed system is characterized by the following variables:

n number of subnetworks in the distributed system;
t_b average backbone link time between two subnetworks;
t_d time for interaction between the EA and the DA;
t_e average time to complete a task in a subnetwork.

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

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

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

$$T_{EA} = (n-1)\,t_b + (n-1)\,t_d. \quad (2)$$

The collector agent time is sum

$$T_{CTA} = (n-1)\,t_b + (n-1)\,t_d + t_e. \quad (3)$$

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

$$T_2 = T_{EA} + T_{CTA} = 2(n-1)t_b + 2(n-1)t_d + t_e. \quad (4)$$

The average time to execute a task in a subnetwork, t_e, is to be considered larger than the average backbone link time, t_b, which depends on the link speed. This is due to some factors. The first is the high-speed technology used for the backbone that allows high transfer data rates. Also, distributed management results in reducing traffic on the backbone and bandwidth saving. The second depends on the small size of the mobile agent transferred on the backbone link. Thirdly, the execution time, t_e, evolves that the DA will visit each node in the subnetwork. The DA delivers the requested task to the multiagent system residing in each element and waits the response. Then the DA transfers from one node to another node in the subnetwork. According to these factors, we can say that t_e is greater than t_b by a reasonable amount and then from equations 1 and 4 we can deduce that T_1 is greater than T_2.

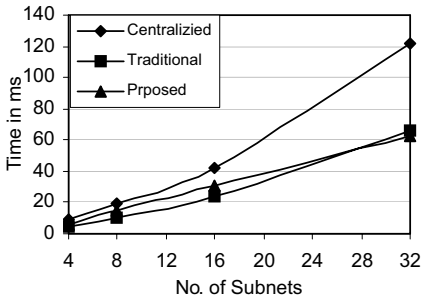
The speedup obtained by the proposed system is calculated as

$$\text{Speedup} = \frac{T_1}{T_2}. \quad (5)$$

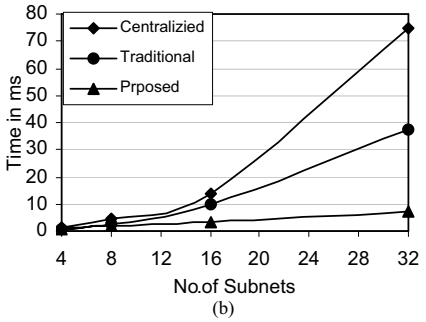
B. Results

The proposed multiagent system is implemented using IBM Aglets [20] 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 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 backbone link speeds such as 10Mbps, 100Mbps, and 1000Mbps. The response time is then compared against both traditional centralized management systems and traditional mobile agent management systems. Speedup is also calculated and reported. This comparison is deployed in Fig. 4.



(a)



(b)

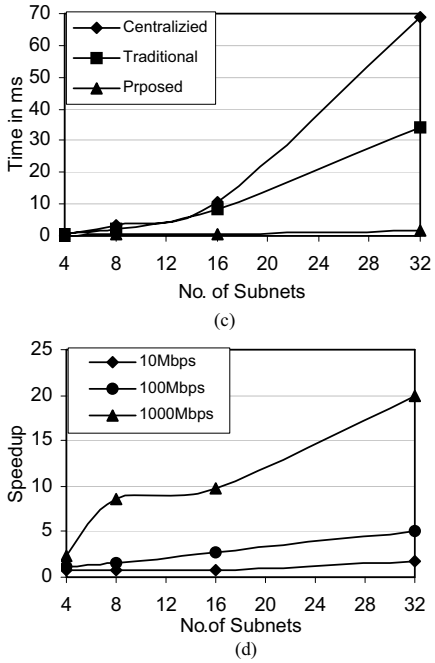


Fig. 4 Response time and speedup. (a) 10Mbps. (b) 100Mbps. (c) 1000Mbps. (d) speedup.

Fig. 4(a), (b), and (c) show that, in the average case, the proposed multiagent system provides response time better than that obtained by the traditional centralized approach and the traditional mobile management systems for different backbone 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(d) 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 traditional management techniques.

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. The proposed system also provides another form of portability called resource portability. It means that the resource can be existed at any node in the system and the proposed system will firstly locate it and then manage it. Thus there is no need to know the location of the resource before managing it.

VI. Conclusions

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 and speedup. A prototype has been implemented using the performance management as the case study. The performance results indicate a significant improvement in response time, speedup, scalability, and portability than the traditional techniques.

References

[1] A. Tripathi, D. Kulkarni, and T. Ahmed, "policy-driven configuration and management of agent based distributed systems," *proceedings of the fourth international workshop on Software engineering for large scale multi-agent systems*, Vol. 30, Issue 4, July 2005.

[2] T. M. Chen, and S. S. Liu, "A model and evaluation of distributed network management approaches," *proceedings of IEEE Journal on Selected Areas in Communications*, Vol. 20, No. 4,May 2002.

[3] A. Westerinen and W. Bumpus, "The continuing evolution of distributed systems management," *IEICE trans. INF. & SYST.*, Vol.E86-D, No. 11, November 2003.

[4] M. G. Rubinstein, O. Carlos, M. B. Duarte, and G. Pujolle, "Improving management performance by using multiple mobile," *proceedings of the 4th International conf. on Autonomous Agents-ACM Agents 2000*, pp. 165-166, Spain, June 2000.

[5] T. C. Du, E. Y. L1, and A. Chang, "Mobile agents in distributed network management," *communication of the ACM*, Vol. 46, No. 7, pp.127-132, July 2003.

[6] G. Coulouris, J. Dollimore, and T. Kindberg, "Distributed Systems Concepts and Design," *Pearson Education*, 3rd Edition 2001.

[7] A. N. Silva, C. A. G. Ferraz, G. L. Ramalho, J. N. Souza, "Proactive network management based on mobile agents and fuzzy logic," *proceedings of IEEE ICT'2001 - International Conference on Telecommunications, Bucharest*, Romania, 2001. V.2, pp. 562 – 566.

[8] C. Raibulet and C. Demartini, "Mobile agent technology for the management of distributed systems – a case study," *proceedings of TERENA Networking conf.*, Portugal, May 2000.

[9] J. Waldo, "Mobile code, distributed computing, and agents," *proceedings of IEE Intelligent Systems*, March/April 2001.

[10] D. G. A. Mobach, B. J. Overeinder, N. J. E. wijngaards, and F. M. T. Brazier, "Managing agent life cycles in open distributed systems," *proceedings of the 18th ACM Symposium on Applied Computing*, pp. 61-65, USA 2003.

[11] S. Kleijkers, F. Wisman, and N. Roos, "A mobile multi-agent system for distributed computing," *lecture notes in computer science*, Vol. 2530/2003, pp. 158-163, august 2003.

[12] C. Huang and C. Pattinson, "Using mobile agent techniques for distributed manufacturing network management," *proceedings of PGNet 2nd Annual conf.*, Liverpool, 2001.

[13] H. Abdu, H. Lutfiyya, and M. A. Bauer, "A model for efficient configuration of management agents in distributed systems," *ACM Performance Evaluation Journal*, pp 285-309, Vol. 54, Issue 4 Dec. 2003.

[14] H. M. Kelash, M. Amoon, G. M. Ali, and H. M. Faheem, "A social agent interface for resource management in distributed systems," *Proceedings of CIMCA*, Vienna, November 2005.

[15] P. Simones, R. Reis, L. M. Silva, and F. Boavida, "Enabling mobile agent technology for legacy network management frameworks," *proceedings of the Softcom'99 – Conference on Software in Telecommunications and Computer Networks*, Split, Croatia, October-1999.

[16] MIB-II of REC-1213, www.ietf.org.

[17] L. R. Welch and D. T. Fleeman, "The ACE orb and distributed resource management," *2nd TAO Workshop*, Virginia , July 2002.

[18] J. Baumann, F. Hohel, K. Rothermel, and M. Straber, "Mole-concepts of a mobile agent system," *World Wide Web*, No. 1, Vol. 3, pp 123-137, 1998.

[19] H. Ku, G. W. R. Ludere, and B. Subbiah, "An intelligent mobile agent framework for distributed network management," *Globecom'97 Phoenix*, AZ, Nov 1997.

[20] D. B. Lange, M. Oshima, "Mobile agents with java: the aglet api1," www.turtle.ee.ncku.edu.tw/~ac/homepage.