Distributed Network Management with Secured Mobile Agent Support

Mohammed Ibrahim
Faculty of Engineering and Information Technology, Taiz University, Yemen

Abstract: *Network computing is changing rapidly these days. The mobile agent technology invented to overcome the complexity resulting due to the increasing size of network components that rises new network management schemes. Many prototype applications providing mobile agent capability have been proposed for being used in network management. E-commerce and information retrieval are some of them. The motive behind the agent mobility is that, it addresses some limitations faced by traditional centralized client-server architecture, which are mainly, minimizing bandwidth consumption, supporting network load balancing, enhancing scalability as well as flexibility, increase fault-tolerance and solve problems caused by unreliable network connections. However, despite its benefits, mobile agent systems still pose security threats. In this paper, we propose a mobile agent architecture that supports flexible and reliable interaction of autonomous components in a distributive network environment. We present a management scheme in a hierarchical level that provides to a user with a reliable and flexible global access to internet/network information services. We further describe a protection mechanism to both agents and their hosting sites of execution called agent servers.*

Keywords: *Mobile agent, domain manager, manager of managers, agent server, agent transfer protocol*.

Received December 9, 2005; accepted April 24, 2006

1. Introduction

Usefulness and viability of mobile agents have been debated since early 90s [2, 3, 12, 20]. Mobile agent technology went through a number of splashes of interest in last 10-15 years. The interest to mobile agents is again on the rise. This rise is motivated by advances in enabling technologies, including wireless networks, diverse.

Broadly speaking, an agent is any program that acts on behalf of a (human) user. A mobile agent, then, is a computer program that is capable of migrating autonomously from node to node, across a heterogeneous network, to perform some computation on behalf of the user [9, 11, 15]. Applications can inject mobile agents into a network, allowing them to roam the network, either on the predetermined path or one that the agents themselves determine based on dynamically gathered information. Having accomplished their goals, the agents can return to their home site to report their results to the user. In managing networks we can achieve more benefits from the use of mobile agents due to the fact that, the approach is based on a decentralized computation. The comparison between mobile agent performance with traditional centralized approach based on Simple Network Management Protocol (SNMP) shows that SMNP does not scale well when the size and complexity of the network increase [4, 13, 14, 16]. In addition, other researchers argue that, to be successful, mobile agent platforms must coexist with, and be presented to the application programmer side-by-side, with traditional client-server platform.

In a mobile agent system, a user simply launches a mobile agent consisting of code and data, and other necessary parameters, to a fixed network, and then disconnects. The agent then navigates autonomously through the heterogeneous networks, interacting with servers or other agents, as it processes the desired information. The mobile agent moves from one server to another while carrying intermediate results. This eliminates the need for the client to maintain a network connection while its agents access and process information. The feature has been proved to be useful for mobile devices, which have unreliable low bandwidth network connections and are often switched off to reduce power consumption. The repeated client-server interactions are therefore reduced to two agents transfer operations, sending a request and notifying the final result. The scenario is illustrated in Figure 1.

In section 2, we present the agent characteristics. Advantages and problems of mobile agents are also discussed in this section. Section 3 tackles the theme of the paper, i. e., proposes the mobile agent management infrastructure and shows how the major functional components are implemented. Fault-tolerance techniques are discussed in section 4. Section 5 presents agent management, agent control and the most critical issue in mobile agent, the security framework. Conclusive remarks are given in section 6. Finally, section 7 concludes the paper.

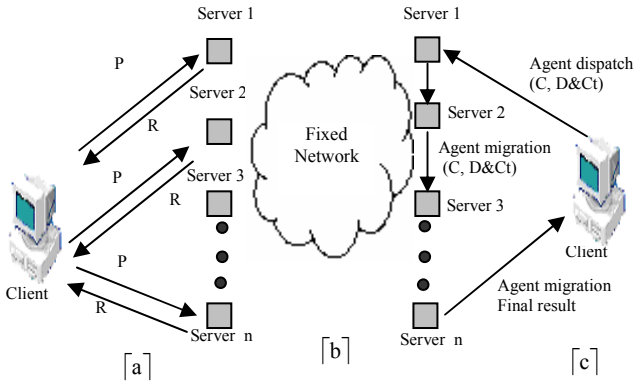

Figure 1. Comparison between SNMP vs. mobile agent paradigms, (a) SNMP paradigm, (b) fixed network, (c) mobile agent paradigm.

2. Agent Characteristics

Mobile agents may possess several or all of the characteristics [1, 6] summarized below in Table 1.

Table 1. Agent characteristics.

Autonomy	Exercise Control Over Its Own Actions
Temporary Continuous	It is a continually running process
Reactivity	Respond in a timely fashion to changes in the environment
Goal Oriented	Does nor simply act in response to the environment
Proactively	Able to change event and make things happen
Social Ability	Can communicate and collaborate
Mobility	Able to transport itself from one machine to another
Cognitive	Able to learn and adapt environment
Flexibility	Characterized by capability to adapt changing environment

Every agent satisfies the first four properties (i. e., reactive, autonomous, goal oriented, and temporary continuos). Other properties are defined on adding hierarchical classification.

2.1. Advantages of Mobile Agents

The advantages of mobile agents are visualized in their significance over the client-server model. Technically there are many mobile agent advantages [10]. These include:

1. Reducing client server bandwidth problems by moving a query from client to the server, this reduces repetitive request/response handshake, as depicted in Figure 1.
2. Solving problems created by intermittent or unreliable network connections. Agents can easily work off-line and communicate their results when the application is back on-line.
3. Supporting parallel execution (load balancing). A large computation can be divided among server's dependent of resources.
4. Allowing decision about the location of code, to be made at the end of the development circle, when more is known about the application, thus reducing the design risk.
5. Providing scalability and robust remote interactions.

All these advantages offer compelling reasons to adapt agent architecture for network management tasks.

2.2. Problems with Mobile Agents

Several mobile agent systems have been proposed [8]. However, the technology is still not yet widely accepted, due to the fact that there are still several issues to be solved. Major problems associated with mobile agent includes:

- *Coordination*: One of the fundamental activities in mobile agent application is coordination between agents and entities they encounter during execution. The mobility of an agent rises problems. Multiple agents are likely to visit the same site at the same time. It is forbidden several agents attempting to have access to the same resource at the same time, as this might lead to deadlock. For the agent-based application to be successful, coordination between agent and network components is an issue that needs to be well addressed.
- *Resource Management*: Since agents are autonomous congestion during resource access is inevitable. Resource allocation of agents must be governed in order to avoid congestion or system breakdown.
- *Security*: The introduction of mobile code in a network rises several security issues. In open network such as Internet, servers run the risks of system penetration by malicious agents, as these can cause undesirable consumption of resources. On the other hand, parts of the agent states might be sensitive and might need to be kept secret when they travel on the network. Security bleach could result in the modification of the agent's code as it traverses the network. Researches admit that protecting the agent against hostile hosts and vice versa are still a difficult issue. Karnik [8] further points out that security is the major obstacle preventing the wide spread acceptance of the mobile agent paradigm. In Section 5.2, we propose a nobel mechanisms for protecting the agent's code and their hosting sites of execution.

3. Proposed Mobile Agent Management Architecture

In this section, we propose an infrastructure that provides framework for network management functionality and code mobility. The architecture emphasizes autonomy and mobility of agents. The entire network is viewed as being made up of small and easily manageable groups of nodes called *domains*. Based on this vision, the infrastructure is equipped

with two major entity categories namely, *management entities* and *managed entities*. The management entities include Domain Managers (DMs) and Manager of Managers (MoMs), all performing the management roles in a hierarchical level. The DMs have control over the nodes within a domain, and MoMs directly exercise power on their immediate subordinate DMs, who in turn exercise the powers delegated to them, to the managed entities.

The managed entities include agent servers, directory servers, service/resource agents and mobile supporting server, each performing specific function. To ease the management, domains consist of servers not more than a certain number. Likewise, the MoMs is bound to serve not more than certain maximum number of DMs. The grouping will largely depend on the geographical layout of the networks and the average load on the entire network. The registration of a node in a group may be done at the time a node is being installed, and a node can be shifted to another domain later as the network administrator may prefer. The mobile supporting server is used to enable the participation of the mobile user in communicating with the network components. The infrastructure is illustrated in Figure 3.

3.1. System Implementation

In the proposed infrastructure illustrated in Figure 2, the managing entities provide an interface to the user to specify policies for mobile agent and dispatch the mobile agent. They also have the capability to create the mobile agent based on the information provided by the user. The travel action plan and security of the management information are specified at the DM before launching the mobile agent. When the mobile agent returns with the collected information, the DM processes the information and presents it to the user in a Graphical User Interface (GUI). In addition, a DM keeps track of the mobile agent, and it is ready at any time to service any special request from any of the managing entities. The same agent code can also be sent to a set agent servers to execute in parallel, thus increasing system performance. Once the mobile agent is launched, the DM is available for other actions such as processing the received results, launching the new agent, etc.

The user at the user interface defines the policies before dispatching the mobile agent. We adopted, with slight modifications, the travelling plan proposed by Wen-Shyen [19]. The travelling plan consists of a list of nodes to be visited, potentially in a specified order. However, the order given at the launching entity is not mandatory; the intelligence of a mobile agent still enables it to make decisions based on the situations at any agent server. Intelligence can also be used in making decisions such as finding the next destination, thus optimizing the travel plan. It can also be used to detect abnormal situations as the agent travels around the network. The security feature, discussed in section 5.2, provides the way to protect the agent, the information collected at the entity, as well as from other agent hosts or entities.

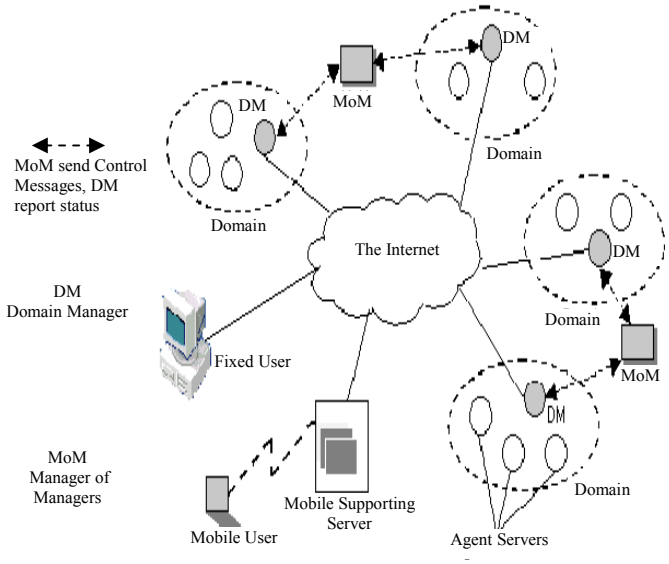

Figure 2. The mobile agent infrastructure.

The agent server is designed to receive and execute mobile code. It thus provides execution environment as shown in Figure 5, and it is responsible for receiving a mobile agent, authenticating the mobile agent and executing under local environment. It must also provide mechanism by which the local resources can be accessed. The agent server program specifies the policies that govern the mobile agent interaction with the local resources. At the same time, the agent server has the authority to deny any service to the mobile agent that violates contract.

The DM strictly specifies interaction between the local environment of the agent server. In order to effectively distribute processing load and control of management station, the hierarchical management approach has been adopted. To keep system manageable, the infrastructure provides only two level hierarchies as shown in Figure 3.

3.2. Implementing Agent Server

The agent server provides the resources, execution environment, as well as communication support for mobile agents. Every node in a mobile agent system must, therefore, be equipped with the agent server. The execution environment consists of Java interpreter called Java Virtual Machine (JVM), which is a stand-alone platform. When the mobile agent arrives in the execution environment with the request to execute the server first performs security operations upon the agent to ensure its safety and regality; it is then instantiated and starts execution. After execution, the server provides storage support for the intermediate results.

An agent server supports the transport mechanism with the help of Aglet Transfer Protocol (ATP). Aglets is a Java system developed by IBM, it enables the agents to move from one server to another by invoking special methods which execute automatically when the agent finishes execution and serving the context in a current host. In the proposed architecture, when the mobile agent wants to move first, the agent server packs it along with the context and encrypt the code for transit protection. When this is done, the method *goTo* executes automatically, and thus enables the agent's code to move to the next destination.

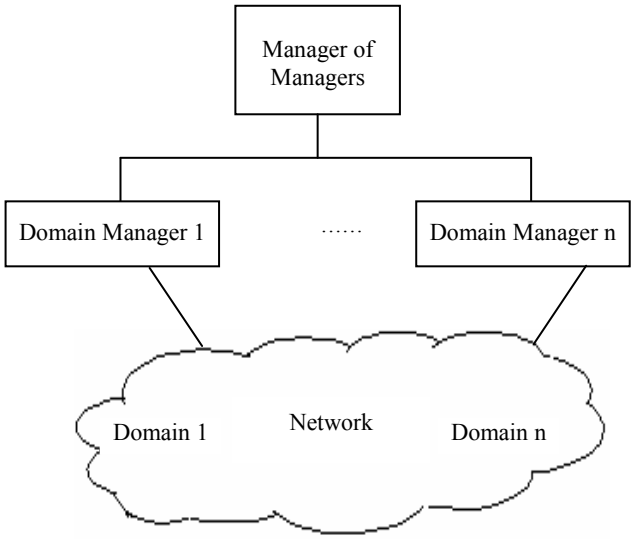

Figure 3. Two level hierarchy network management.

3.3. Mobile Agent Life Circle

In the proposed architecture, a user creates the agent containing the request, mobile code, state information and other parameters. Other attributes include information about the mobile agent, such as the launcher, movement history, resource requirement, identification, authentication and encryption information. After creation, the user consults the manager for the agent launch. After arriving at the new host, a mobile agent performs security and initializing process to activate itself. Once activated, the mobile agent collects the necessary information through interaction with local resources and performs execution. After the completion of all the specified tasks for that particular entity, the results are served into the container of the mobile agent. The agent will continue visiting nodes one after the other until the assigned task is completed. Figure 4 describes the mobile agent journey in its lifetime.

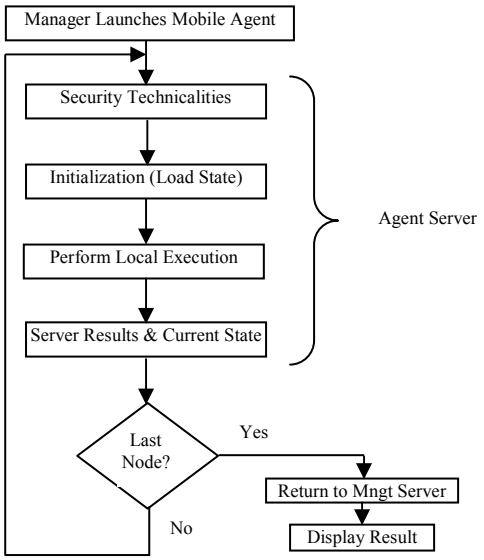

Figure 4. Mobile agent life circle.

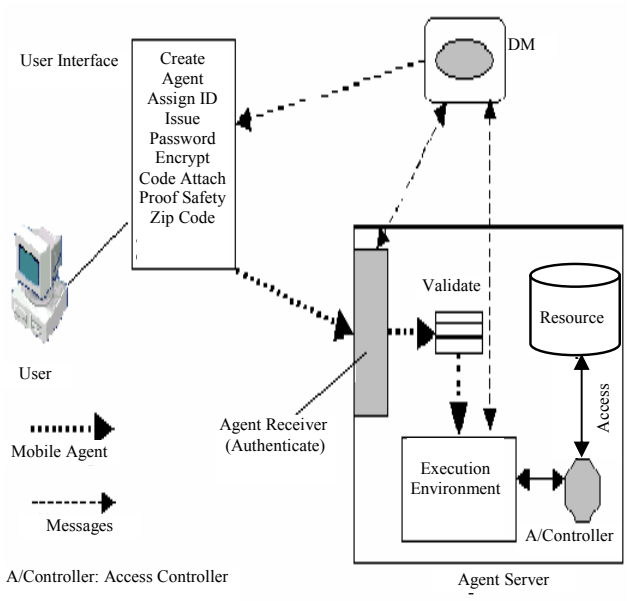

Figure 5. Mobile agent security environment.

4. Fault-Tolerance

A MoM is responsible for coordinating the DMs requesting information and delegating tasks to them. In order to provide fault-tolerance, DMs may perform functions of a neighbor manager as it may be instructed by MoMs. MoMs keeps watch on the DMs and immediately delegates power in case of failure. DMs may also communicate with each other to perform some management tasks and analyze network in a global manner. A mobile agent is sent from the DM to the agent server to carry out the task locally, dynamically delegating management functions. In case of any execution problems or coordination problem which cannot be attended by the by the DM, a message is sent to the MoM to seek help. On the other hand, in case of misbehavior in a mobile code, a message is sent to the DM and dynamic code change can be

performed, to send another mobile agent as a substitute to the old one.

5. Agent Management

For the mobile agent to be useful, a management scheme is necessary. Managing an agent involves monitoring its events and actions. The mobile agent events include agent creation, migration, arrival, termination, and agent interaction with other agents. The key issues in agent management are agent monitoring and agent control, which in most cases rise the security issue. In this section, we briefly discuss control techniques and concentrate on security issues. The management server needs to hold the status of the mobile agent and provide control functions to the user. For all these to be possible, the management server needs to have a way to first locate the mobile agent.

5.1. Agent Control

The agent application might need to monitor the agent's status while executing. If exception or errors occur during execution, the application needs to terminate the agent, which involves tracking the agent's current position and requesting the host server to kill it. Similarly, the agent launcher (or user) might want to call it back to its home site and allow it to continue execute there or terminate it for some reasons.

In the proposed system, the application or user invoke the primitive *retract* to call the agent back. To be able to have control over the agent, the first action to take is to locate it. The proposed system adopted the hierarchical agent location tracking mechanism proposed by Jonathan Dale [17]. Every agent server creates a path registration for each mobile agent as it passes through the server. The path consists of the pointer to the next hope where the agent was last transferred and thus forming a unique path from the root node of the tree down to agent running environment. The messages being delivered to the agent easily follow the same route as the agent until they find it.

5.2. Security System Design

As mentioned in section 2.2, not all hosts through which agent passes are trustful. Deliberate measures for protecting mobile agent, sensitive data and mobile hosts are therefore necessary. However, it must be noted that the security threats in mobile agent systems come not only from agent hosts, but also from malicious agents. Malicious mobile agents may compromise and/or modify sensitive data, which they have no right to access; can also interfere with the execution of other agents. On the other hand, malicious hosts may cheat the agent migrating to them and therefore interfere with the successive execution of the mobile agents. Having this obvious assumption in mind, we propose an agent security mechanism that takes effort to protect both the hosts and the mobile agent as illustrated in sections 5.2.1 and 5.2.2.

5.2.1. Agent Server Security Mechanism

Before the agent server can allow the mobile code to execute in its environment, it verifies that the agent is suitable for execution by enforcing *authentication*, *verification* and *authorization* security techniques.

As soon as the mobile agent arrives at the agent server, before it can perform any task, it is authenticated. The agent server asks for the *authentication* details. In fact, after creation at the user's terminal, the agent registers itself at the DM, and it is assigned a password. Before the agent is accepted at any host, it is required to supply a password, its ID and the mobile agent launcher's ID as shown in Figure 5. A mobile agent that fails authentication is rejected by the agent server.

Verification involves checking the mobile agent code to ensure that it does not perform any prohibited actions. To achieve verification process, a technique called proof caring code [17, 18] was adopted. Each code to be transferred; a safety proof is attached to it. Once the mobile agent arrives at the new host, the agent server validates the proof to each piece of code and thus ensures that the code is safe to execute.

Authorization deals with agent access permission to the agent server resources, such as PCU circles, read/write access, etc. In our proposed system, only trusted agents can read, write and do modifications. In fact, less trusted agents have limited access to resources. The access controller provides permission to the mobile agent to access resources, and is able to determine trusted and non-trusted agents, and provide services accordingly.

5.2.2. Mobile Agent Security Framework

In Figure 5, we illustrate the mobile agent framework. The areas for security checks are agent receiver, security manager (or validate machine) and the access controller. Hosts also need to be authenticated to check their safety before mobile agents' moves to them. In protecting the software code of the mobile agent as well as information at different agent servers, cryptographic algorithm were adopted, Pretty Good Privacy (PGP) encryption method based on RSA DEA and IDEA.

The motive behind adopting PGP encryption method was that, it provides not only mobile security but also code compression. In addition, the complete package, including all the source code, is distributed free of charge via the Internet, and it is available on MS-DOS/Windows and INIX platforms. After the code is encrypted, it is then zipped using the ZIP program developed by Ziv and Lampel [22], it is then

launched. At any time, the DM keeps watch on the mobile agent movements.

The mobile agent code together with its data is encrypted before being launched. Code encryption ensures that the agent will reach the next destination safely. Only the agent execution environment specified in the itinerary, and that holds the agent decryption key, will be able to execute the agent [5]. The execution environment consists of a JVM code, which provides the platform for the agent execution. Any entity from the outside environment cannot interfere with the execution environment. If it is necessary to interfere, it does so through a restricted interface that is controlled by the validate module, which also guarantees that the agent code will be executed correctly.

6. Performance Assessments

We assess the effect of providing confidentiality and integrity as an effort to protect a mobile agent-based management system from malicious hosts attack. In our approach, we develop a simple but reasonable analytical model based on performance parameters considered to be the three main parts of a mobile agent-based management system; i. e., the agent code, the agent transfer protocol APT, data and results obtained at each visited host. While the agent code and the APT are assumed to immutable, data and result are expected to vary. The size of a mobile agent-based management system is expected to increase as a mobile agent-based management system collection on each visited host.

In our model, we consider data to be the input required by the agent to perform computation at a particular hosting site; the agent owner usually supplies data before the agent is launched, more data may be added as needed from system resources such as directory servers. We also consider results to be the amount of information collected at any visited site. The user sends a mobile agent-based management system to the network/Internet. The mobile agent-based management system then visits the hosts one after another as specified in the itineraries and when satisfied, returns to the user to submit results.

The mobile agent-based management system preserves the agent integrity at the expense of execution over head. The delay on a mobile agent-based management system is mainly due to two factors:

1. Time spent on results encapsulation and cryptography computations at the intermediate sites.
2. Delay of the sender for verifying the agent integrity, we define the total delay $D\tau$ introduce d as a result of implementing the mobile agent-based management system as follows:

$$D\tau = ND\varepsilon + ND\upsilon \quad (1)$$

Where $D\varepsilon$ denoted the delay in performing encapsulation and cryptographic computations at any given host, $D\upsilon$ denoted the delay introduced by the sender for verifying the agent integrity and N is the total number of hosts visited. As mentioned in the earlier sections, the encapsulation scenario involves cryptographic computation, performing the harsh on the partial results and signing the entries. With this consideration $D\varepsilon$ becomes:

$$D\varepsilon = D\eta + D\kappa + D\delta + D\sigma \quad (1\text{-}1)$$

When $D\eta$ is the delay introduced in computing the harsh, $D\kappa$ and $D\delta$ are delays due to encrypting and decrypting the agents and $D\sigma$ is the signature processing delay.

The sender verification on the code integrity involves creating a new harsh using the original code and making a comparison, the sender also verifying the signatures. With this notion, $D\upsilon$ can be defined as:

$$D\upsilon = D\eta + D\kappa + D\upsilon\sigma \quad (1\text{-}2)$$

Where $D\upsilon\sigma$ is the delay introduced by the sender for verifying signatures. With 1-1 and 1-2, we have:

$$D\tau = N(D\eta + D\kappa + D\delta + D\sigma) + N(D\eta + D\upsilon\sigma) \quad (1\text{-}3)$$

Assuming that $D\sigma \equiv D\upsilon\sigma$ we have:

$$D\tau = 2N(D\eta + D\sigma) + N(D\delta + Dk) \quad (2)$$

For the overall system performance, the transmission overhead is very significant and should not be neglected. The parameters for transmission delay are code transfer; data/result transfer and the atp transfer. The transmission delay between two successive hosts is then:

$$Dij = (D\varphi + D\rho + D\gamma) \quad (3)$$

Where Dij denotes the transmission delay between two successive hosts I and j, $D\varphi$ denotes the delay introduced by code transmission, $D\rho$ denotes the delay introduced by the ATP and $D\gamma$ is the delay introduced by the data/results transfer.

Transmission delay is the function of the size of the mobile agent-based management system. Since the code and the ATP are immutable, the varying parameter in equation 3 is the third term $D\gamma$. We expect the size of the agent to grow as it gathers information from the visited hosts. Assuming that the amount of information collected by an agent on each visited site is the same; the total delay due to mobile agent-based management system transmission may be expressed as:

$$D\tau = NDij = (D\varphi + D\rho) + \sum_{0}^{N} \Delta D\gamma) \quad (D\varphi + D\rho) + (N+1)/2\ \Delta D\gamma \quad (4)$$

Where ΔD is the size of information obtained at the visited host.

7. Conclusion

In this paper, a mobile agent-based management system has been discussed. The framework differs from most other mobile agent frameworks in that, it proposes a hierarchical level of management, which provides, to network components, smooth coordination and fault-tolerance mechanism. It also provides an efficient way of locating the agent. This is achieved by having the agent creating a registration path as it passes though the servers. The approach enables the message being sent to trace the agent, easily follow the path and go directly to the agent.

Since mobile agents are autonomous, security becomes an important issue. This work implements security mechanisms that protect not only the agent but also the agent hosts. A Pretty Good Privacy (PGP) encryption method has been adopted to protect the agent code from being altered as it traverses the network components.

Agent-based network management are not yet widely accepted due to the fact that, most of the systems proposed so far still suffer security and coordination problems. However, a lot has been done by researches as mobile agent systems seams to solve many problems existing in the traditional client-server model. It is our hope that agent-based interaction will become an important paradigm for the future network management systems. In addition, the possibility of combining mobile agent interaction with SNMP interaction gives the possibility of coming up with a most robust and efficient applications.

References

[1] Abeck S., Koppel A., and Seitiz J., "A Management Architecture for Multi-Agent Systems," *in IEEE Proceedings of the 3rd International Workshop on System Management*, Boston, USA, pp.133-137, April 1998.

[2] Abowd G. D. and Mynatt E. D., "Charting Past, Present, and Future Research in Ubiquitous Computing," *ACM Transactions on Computer-Human Interaction*, vol. 7, no. 1, pp. 29-58, March 2000,.

[3] Arcos J. and Plaza E., "Exploiting Context Awareness in Information Agents," *in Proceedings of the 5th International Conference on Autonomous Agents, Montreal*, Canada, pp. 116-117, 2001.

[4] Baldi M. and Picco G. P., "Evaluating The Tradeoffs of Mobile Code Design Paradigms in Network Management Applications," *in Proceeding of the 20th International Conference on Software Engineering (ICSE'98)*, Kyoto, Japan, pp. 146-155, April 1998.

[5] Brewington B. and Gray R., *Mobile Agents for Distributed Information Retrieval: Intelligent Information Agents*, Springer-Verlag, 1999.

[6] Buchman W. J., Naylor M., and Scott A. V., "Enhancing Network Management Using Mobile Agents," *IEEE Transactions on Knowledge and Data Engineering*, vol. 12, no. 5, pp. 818-826, 2000.

[7] Jonathan D., "A Mobile Agent Architecture for Distributed Information Management," *PhD Thesis*, University of Southampton, September 1996.

[8] Karnik N. and Atripathi, "Agent Server Architecture for Ajanta Mobile-Agent Systems," *in Proceedings of the International Conference Parallel and Distributed Processing Techniques (PDPTA'98)*, Las Vegas, USA, pp. 63-73, 1998.

[9] Karnik N. M. and Tripath A. R, "Design Issue in Mobile Agent Programming Systems," *in Proceedings of the IEEE Concurrence*, Boston, USA, pp. 52-61, July-September 1998.

[10] Lange D. B. and Oshima M., "Seven Good Reasons for Mobile Agents," *Communications of the ACM*, vol. 42, no. 3, pp. 355-395, 1999.

[11] Loke S. W. and Zaslavsky A., "Mobile Agent Supported Cooperative Work: The ITAG Scripting Language Approach," in Ye Y. and Churchill E. (Eds), *Agent Supported Collaborative Work*, Kluwer Academic publishers, 2002.

[12] Loke S. W., Padovitz A., and Zaslavsky A., *Context-Based Addressing: The Concept and an Implementation for Large-Scale Mobile Agent Systems Using Publish-Subscribe Event Notification*, Lecture Notes in Computer Science, vol. 2893/2003, Springer Berlin/Heidelberg, France, November 2003.

[13] Marcelo G. R., Otto C. M., and Duarte B., "Evaluating the Performance of Mobile Agents in Network management," *in Proceedings of the IEEE Global Telecommunications Conference*, Rio de Janeiro, pp. 386-390, December 1999.

[14] Pangureck Y., Wang Y., and White T., "Integration of Mobile Agents with SNMP: Why and How?," *in Proceedings of the IEEE Symposium on Network Operations and Management*, Jeju Island, Korea, pp. 609-622, September 2004.

[15] Papastavrou S., Samaras G., and Pitoura E., "Mobile Agents for World Wide Web Distributed Database Access," *IEEE Transactions on Knowledge and Data Engineering*, vol. 12, no. 5, pp. 802-820, 2000.

[16] Ravi J., Farooq An. and Amjad U., "*A Comparison of Mobile Agent and Client–Server Paradigms for Information Retrieval Tasks in*

Virtual Enterprises," *in IEEE Proceedings*, Jeju Island, Korea, pp. 302-313, September 2002.

[17] The Official PGP User's Guide, Cambridge, MA MIT Press, 1995a.

[18] The Official PGP User's Guide, Cambridge, MA MIT Press 1995b.

[19] Wen-Shen E., Lin C.C.Y., and Lien Y. N., "A Mobile Agent Infrastructure with Mobility and Management Support," *in Proceedings of the IEEE International Workshop on Parallel Processing*, Wakamatsu, Japan, pp. 508-513, September 1999.

[20] Zaslavsky A., "Mobile Agents: Can They Assist with Context Awareness," *in Proceedings of the 2004 IEEE International Conference on Mobile Data Management (MDM'04)*, California, USA, January 2004.

[21] Zaslavsky A., "Mobility in Enterprise Applications," *in Proceedings of the 5th International Conference on Business Information Systems*, Poland, April 2002.

[22] Ziv J. and Lampel Z., "A Universal Algorithm for Sequential Data Compression," *IEEE Transactions on Information Theory*, vol. IT 23, pp. 337-343, May 1977.

Mohammed Ibrahim received his BSc degree in computer science from Baghdad University, Iraq, in 1991, MSc in computer sciences & engineering from Shanghai University, China, 1996, and his PhD degree in computer sciences and engineering from Shanghai Jiaotong communication University, China, 2003. His research interests include grid computing and distributed systems.