December 2003

Pool-Based Heterogeneous Mobile Agents for Network Management

Nandini Taneja
InfoVision Consultants Inc.

Aakash Taneja
University of Texas at Arlington

POOL-BASED HETEROGENEOUS MOBILE AGENTS FOR NETWORK MANAGEMENT

Nandini Taneja
InfoVision Consultants Inc.
nandini_taneja@hotmail.com

Aakash Taneja
University of Texas at Arlington
aakasht@hotmail.com

Abstract

The use of mobile agents to manage networks provides an efficient paradigm shift from traditional client/server architecture. Mobile agents provide an intelligent automatic management of wide network nodes consuming less computer and network resources. In this paper, we propose a prototype system with a heterogeneous mobile agent pool. We introduce a heterogeneous agent pool – where different type of agents co-exists. The pool is configurable to start a predefined number of mobile agents of each type. The set of stubs deployed with the agent determines the task performed by them. Agent pooling is an efficient way to manage resources and control the number of mobile agents in the network. It also makes the system scalable. In this paper, we propose an implementation of Network Node Fail Over scenario with our prototype system.

Keywords: Agent pool, element management, mobile agent, network management, SNMP

Introduction

The management of networks that is mostly centered in a Network Operations Center (NOC) is done by a Network Manager using several Element Management Systems (EMS) to manage their specific zones or domains (Figure 1). These EMS need to adhere to the ISO's five network management categories: accounting, security management, configuration, provisioning, and fault detection (Koegh 2000) to manage the network elements. The protocols used to manage the nodes are either *Simple Network Management Protocol (SNMP) defined in RFC* 1157 (Case et al. 1990) *or Common Management Information Protocol (Stallings 1999). Since a*n EMS manages a specific portion of the network with complex multiple network devices, it needs to have a stable architecture to handle scalability, resilience, and heavy loads.

We have based our system on SNMP (Stallings 1999) protocol since its simplicity makes it widely used to enable network administrators to manage network performance, find and solve network problems, and plan for network growth. The three categories of operations (Stallings 1999) that can be performed on network devices using SNMP are:

- Retrieving Data – get, getnext, getbulk
- Configure or change variables using – set
- Receiving messages from devices using – traps

SNMP is based on manager-agent model where manager send commands to agents to retrieve data or configure its variables. The network nodes that use SNMP have agent software to communicate with any manager. The agent spontaneously sends information to the manager as traps. Each piece of information to be collected about a network node or device is defined in a "Managed Information Base" (MIB) (Stallings 1999).

Network Management Systems have been based on client/server architecture where manager requests information from agents. In Figure 1, Network Manager acts as a client to the EMS, which in turn acts as a manager to network elements to retrieve and configure data on network nodes. Since EMS manages many complex network devices with high volume of data traffic, it thus needs to be capable of managing high loads, be fault tolerant and scalable. This client/server approach consumes lots of network and CPU resources, and the bandwidth required for the application, thus degrading the overall performance. Shu et al. (2000) have discussed various approaches to move away from traditional client/server paradigm towards an agent-based architecture.

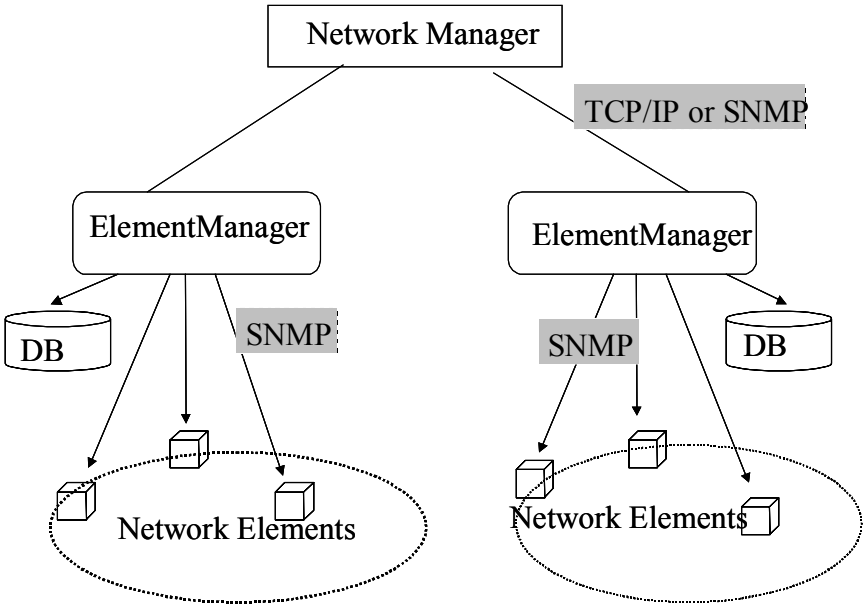

Figure 1. Network Management Architecture

A mobile agent is a piece of code that can migrate from host to host in a network. It can be triggered either by an agent or on its own. Since an agent migrates after performing its task, it improves system performance and reduces network load (Michalas et al., 2001). Covaci et al. (1997) discuss the use of intelligent mobile agents to manage the network devices. Shu et al. (2000) discuss mobile agents to manage networks using CORBA (Common Object Request Broker Architecture).

In this paper, we propose a prototype system with agent pool having different type of mobile agents that co-exist at the same time. Our approach uses simple timeout mechanism to control the number of mobile agents roaming in the network. These mobile agents are capable of performing well-defined tasks. We restrict two similar agents to perform a simultaneous SNMP SET operation on a network element to avoid redundant operation.

Our pool concept originates from the idea used by many application servers today to have object pools to handle requests. The main benefit of pooling is that once the objects are created in a pool, they are not required to be recreated, thus saving creation time and enforcing proper startup and shutdown of these objects. The advantages to use this approach in network management as compared to the earlier work are:

a. In a SNMP based EMS, fault management has been done either by polling the Network Elements (NE) or by receiving faults as traps sent by NEs. It is not possible to balance the load in the EMS dynamically by changing the number of fault modules or other modules in case of any need. In our approach, more than one type of mobile agent can exist in the agent pool and system can be configured to load balance such that if management system needs to do more fault related diagnosis, it can have more fault-type mobile agents and less of other agents (e.g.: less configuration-type mobile agents).

   The use of agent pool provides a control on the number of mobile agents roaming in the network, thus reducing network traffic, CPU load, system memory consumption and increasing performance. It helps to save initial startup and agent initialization time and provides graceful startup and shutdown of agents. At any given time, the number of agents of a particular type that are busy performing the tasks can be found from the agent pool. After the agent completes its task, the knowledge base is updated with the response from agent and the agent is reallocated to the pool. More than one agent of same or different type can be simultaneously marshaled in the network. In our approach, we have used agent timeout mechanism to re-marshal similar agent if the earlier one does not come back due to network delay or loss of agent itself.

b. Zhang and Sun (2001) have discussed the presence of JVM on each NE for the mobile agent to visit the nodes for diagnosis. In our approach, there is no need to have any additional software on the NE. Thus, old legacy network elements can be monitored using this architecture.

c. Our approach is scalable such that any new mobile agent with appropriate stubs can be configured and uploaded to the system.

System Architecture

We propose an EMS based on mobile agents where the Base Agent (BA) is a heavyweight intelligent agent and is the mastermind of the whole system. BA can receive SNMP packets from the Network Manager. Since BA acts as an agent for outside network managers and as a manager for all the network devices it manages in its domain, it needs both stubs and skeletons. It obtains stubs and skeletons by compiling the MIBs. There are many SNMP based MIB compilers available today to compile the MIBs. BA plugs respective stubs to mobile agents, thus providing them heterogeneity. An example is the MIB that generate the fault stubs for the fault diagnosis mobile agents. BA also sends instructions to mobile agent to perform operation on one or multiple network elements.

The Base Agent as shown in Figure 2 maintains a powerful agent pool to keep sets of different type of mobile agents during startup. The number of each type of mobile agent can be configured to a MINIMUM and MAXIMUM. Following are some of the heterogeneous mobile agents in our system – Fault Management Agent, Configuration Agent, Provisioning Agent, Software Download Agent, etc.

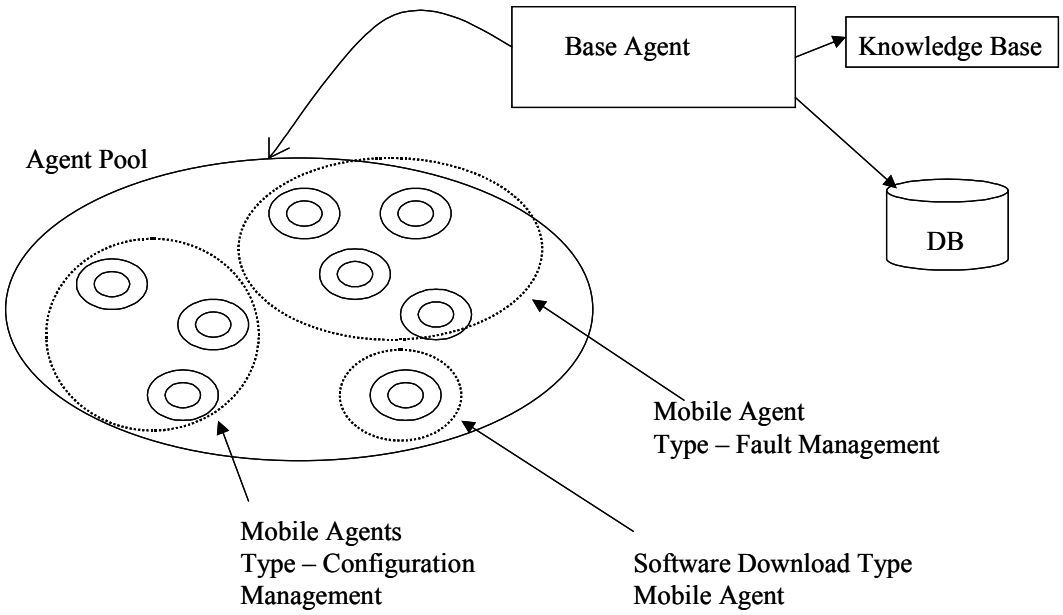

Figure 2. Base Agent Architecture

Since BA needs a platform to marshal its mobile agents, we introduce Agent Proxy that acts as a landing pad for the mobile agents to perform SNMP operations on the network element. The Agent proxies are launched by the BA during the startup and have a one-to-one relation with network elements in our system. So for each element there will be an Agent Proxy, resulting in many Agent Proxies in a network.

Agent Proxy has an information base with information about the network element where it is launched. e.g. Network element's IP address, backup node, software release on it and the last time it was upgraded, etc. It has all the functionality to receive a mobile agent, provide environment for its execution, migrate the code to another Agent Proxy (if required) and avoid duplicate execution of a mobile code (in case base agent re-marshals an agent during timeout). Agent Proxy does not allow two mobile agents of same type to perform simultaneously on a NE. Agent proxy maintains a queue where each mobile agent enters and does a handshake to check the agent's type and operation before starting it. (Figure 3)

The Mobile Agent is marshaled by Base Agent either on-demand (e.g., when a request for configuration or provisioning a network node comes from network manager) or when Base Agent decides to do diagnosis or resend an agent during timeout scenario. The Mobile Agent lands at the Agent Proxy, does the SNMP based get/set operations, and migrates back to the Base Agent. Before adding the Agent back to the agent pool, BA gathers the diagnosis information along with results of the operation on the NE and updates its knowledge base.

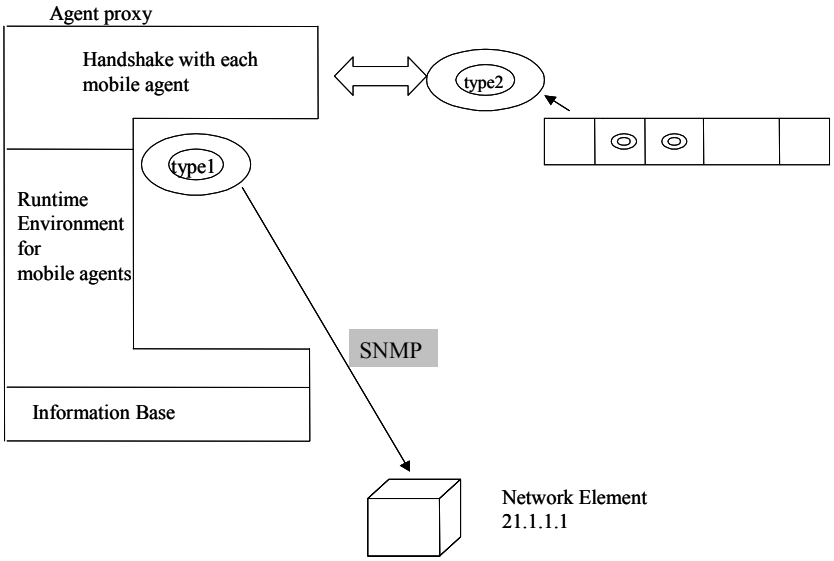

Figure 3. Agent Proxy and Mobile Agent Interaction

System Implementation

Base Agent

The Base Agent reads a list of configuration parameters as an attribute value pair to determine the startup of various mobile agents or agent proxies. The information related to agent proxy is stored in EMS database whenever it is updated or modified while the system is running. Following are some of the configuration parameters for BA:

```
# Configuration File for Base Agent
# Following pairs determines the (MINIMUM, MAXIMUM) number of mobile agent that will be started by Base Agent
FAULT_AGENT=4,7
CONFIGURATION_AGENT=3,6
SOFTWAREDNLD_AGENT=1,2
#Following section lists the Workstations where Base Agent can launch the agent proxies
AGENT_PROXY_RANGE=113.128.142.72,113.128.128.28-40
AGENT_PROXY_PORT=1025-10000
```

The Base Agent launches an Agent Proxy on the available workstation's respective port and stores this information in its database. It also has a Knowledge Base that uses rules to determine any change in the system, or diagnosis of the system and updates the database. The number of mobile agent of each type depends on the "MINIMUM" and "MAXIMUM" value as specified in the configuration file. If due to network congestion the mobile agents of a particular type reach the MAXIMUM limit, the Base Agent does not start any new mobile agent of that type. This makes the administrator aware of the network / system issue that needs to be resolved.

Each mobile agent is plugged with stubs to perform respective network management operation. These stubs are Java codes that are generated by compiling MIBs using any MIB compiler like SUN's mibgen compiler for Solaris (available at **http://www.sun.com/products-n-solutions/telecom/software/java-dynamic/overview.html**). Mibgen is a part of Java Dynamic

Management Kit; a Java based toolkit from SUN for developing agents. Mibgen can be used to generate stubs or both stubs and skeletons from the MIBs. Each type of agent is a Java class that has respective stubs plugged to it.

Agent Proxy

As soon as the Base Agent gets a new NE to be configured and managed, it launches an Agent Proxy for that NE. The Agent Proxy gets all the information about the NE from Base Agent that gets it from the manager as a part of configuring the NE. Some of the information on the NE can come later after it starts up. All this is updated at Base Agent and corresponding Agent Proxy for the NE. Agent Proxy has a queue where it receives the mobile agent code. It reads each agent code one by one from the queue and performs simple handshake. Once the handshake is complete, the agent proxy provides environment for the agent to run and communicate with NE using SNMP operations. The handshake between the mobile agent and agent proxy does a basic username/passwd authentication and determines the type of mobile agent and its operation, the current state of the mobile agent and timeout for the mobile agent.

Network Node Fail Over Scenario – An Example

In this section, we show how our approach can be used for fault management and perform a network nodes fail-over. Most network nodes run applications that are time critical and if it fails, its switch over to backup should be efficient and fast so that no data is lost and it is transparent to outside world.

As discussed above, a NE based on SNMP protocol can be configured to send traps to the manager. These traps hold important information that determines the type of fault, its severity level, the time of occurrence, IP address of node, etc. Zhang and Sun (2001) discuss a mobile agent based approach for fault detection where mobile agents roam around the network gathering information from every network component (NC) and reporting the results to the manager. El-Darieby and A.Bieszczad (1999) discuss an intelligent agent based approach where the mobile agent has the expertise to diagnose the faults locally in the NC and detect a pattern of faults.

In our approach, instead of roaming in the network, the Base Agent marshals mobile agents either when needed or when asked by the network manager. The Base Agent receives all the faults as SNMP traps from network elements and knows exactly when to marshal a mobile agent when a node fails. The failure of a node can be detected based on the traps received from the NE; e.g., NE could be sending some critical level traps on failure of its sub-systems before it finally goes down. Once the BA determines to switch over a node to its backup, it marshals its respective mobile agent to perform the operation.

We consider network element “NE1a” as Master and its backup network element “NE1b” as Slave (Figure 4). When NE1a fails, BA marshals the mobile agent at Agent Proxy queue for NE1a. Agent Proxy performs quick handshake and lands the agent. The mobile agent gathers information from Agent Proxy (IP address, software version for slave node, etc.) and configures to bring it up. We assume that the slave node is at a state such that it has to be configured to become active in sending traps to the BA. Thus, slave node is booted, but is in an inactive state. The mobile agent then changes the state of NE1b to active such that NE1b becomes master. Mobile agent then communicates with the failed node, tries to bring it up in an inactive state; collect all the diagnosis information and its current state from failed node, and migrates back to the BA. BA then updates its knowledge base and database.

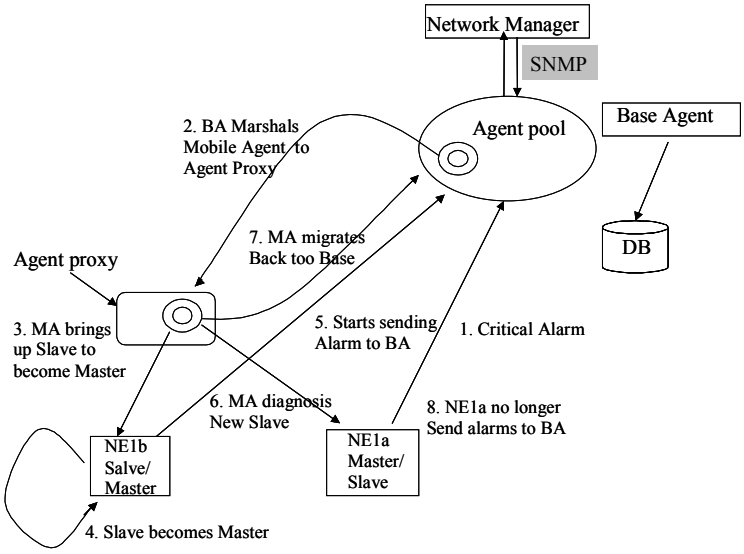

Figure 4. Network Node Fail Over Scenario

Contribution

In this research under progress, our contribution lies in proposing a heterogeneous mobile agent pool that provides an efficient, fault tolerant and scalable EMS solution. As an interest to practitioners, the use of agent pool provides a control on the number of mobile agents roaming in the network, thus reducing network traffic, CPU load, system memory consumption and increasing performance. It helps to save initial startup and agent initialization time and provides graceful startup and shutdown of agents. More than one agent of similar or different types can also be marshaled in the network at the same time.

Conclusion and Future Work

In this approach, we have based the system on SNMP although it can be implemented for any protocol like CMIP, TCP/IP with MIBs available to generate the stubs for the Base Agent. Since Non-SNMP systems do not support trap mechanism for fault management, the Base Agent can be changed to poll the nodes instead of receiving traps. We intend using Java for implementation because of Java's interoperability and platform independence. In addition, Java comes as a handy language because of its ease to migrate code, Remote Method Invocation and JVM.

In our system, we have only considered the possibility of mobile agents performing operations using one type of management protocol. However, this approach can be extended to a bigger agent pool with agents co-existing with each other and capable of performing SNMP, TL1 operations on different type of network nodes. Since Java is an interpreted language and has performance limitations, we plan to use UNIX (Kernighan and Pike 1984) system's crontab utility and good old File Transfer Protocol- ftp (Postel and Reynolds 1985) to implement the migration of agents. This will help to use any language to develop the system and is our interest for future research.

References

Case, J., Fedor, M., and Schoffstall, M. " A Simple Network Management Protocol (SNMP)," RFC1157, May 1990.

Covaci, S., Zhang, T. and Busse, I. "Java-based intelligent mobile agents for open system management," Proceedings of the Tools with Artificial Intelligence, 1997. Proceedings., Ninth IEEE International Conference on, 1997, pp. 492-501.

El-Darieby, M., and A.Bieszczad "Intelligent mobile agents: towards network fault management automation," Proceedings of the Integrated Network Management, 1999. Distributed Management for the Networked Millennium. Proceedings of the Sixth IFIP/IEEE International Symposium on, 1999, pp. 611-622.

Kernighan, B.W., and Pike, R. *The UNIX Programming Environment*, Prentice Hall Professional Technical Reference, New Jersey, 1st edition 1984.

Koegh, J. *The Essential Guide to Networking,* Prentice Hall PTR, New Jersey, 1st edition, 2000

Michalas, A.K., T.; Kalogeropoulos, S.; Karetsos, G.; Sidi, M.; Loumos, V. "Enhancing the performance of mobile agent based network management applications," Proceedings of the Sixth IEEE Symposium on Computers and Communications, 2001, pp. 432-437.

Postel, J., and Reynolds, J. " File Transfer Protocol (FTP)," RFC959, Oct 1985.

Shu, T., Yang, C., and Jianhua, Y., Ning Y "A mobile agent based approach for network management," Proceedings of the International Conference on Communication Technology, 2000. WCC - ICCT 2000., 2000, pp. 547-554 vol.1.

Stallings, W. *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*, Addison-Wesley, 1999.

Zhang, P., and Sun, Y. " A new approach based on mobile agents to network fault detection," Proceedings of the IEEE International Conference on Computer Networks and Mobile Computing, 2001, pp. 229-234.