Unifying WBEM and Mobile Agents Approaches for Systems Management

Marco Antonio C. Simões[1,2], André Luis de M. Santos[1]

[1]Centro de Informática – Universidade Federal de Pernambuco
P.O. Box 7851, ZIP: 50732-970 – Recife–PE – Brazil

[2]Faculdade Integrada da Bahia
Rua Xingu, 179 – Jardim Atalaia/STIEP – ZIP: 41770-130 – Salvador-BA – Brazil

e-mails: {macs3,alms}@cin.ufpe.br

Abstract

Systems Management (SM) must be able to manage all aspects of computers systems, from physical details of hardware and communication to applications, including communication protocols and middleware. To meet these needs, several approaches based on intelligent mobile agents have been proposed. The rapid evolution of computer systems provides an additional issue for SM systems: the capability to support new types of managed objects in a flexible and extensible way. To match this need, IT industry has provided a new initiative, WBEM (Web-Based Enterprise Management). In this work, we validate the application of SM architectures based on intelligent mobile agents with standards proposed by the WBEM initiative. To reach this goal, we developed a framework to support intelligent mobile agents capable of accessing management information according to WBEM standards.

Keywords: Network and Systems Management, CIM, WBEM, Mobile Agents

1. Introduction

Computer networks are currently used to support many distributed applications that differ from traditional client-server model. The use of diverse architectural models [Coulouris et al 2001] such as mobile code, distributed events, distributed objects and peer-to-peer has created new management demands that are not supported by traditional network management tools.

This scenario highlights the limitations of classical network management architectures. Low scalability and high management traffic in the network are the main problems on these traditional solutions.

An alternative paradigm that has been evaluated in the last few years is the use of mobile agents for systems management. This approach consists of using small software applications capable of moving between network devices. This software has autonomy and reasoning capabilities to perform tasks they were previously programmed to execute. These processing elements are called intelligent mobile agents. The main idea is to move parts of the management application code to the location of the data it will use, instead of moving the data to the location of the management application code, as in client-server paradigm.

The emergence of mobile agent frameworks has led many researchers to examine their applicability to network and systems management. Bieszczad et al [Bieszczad et al 1998] discussed the general issues of using mobile agents for network management. Rubinstein et al [Rubinstein et al 2000] compared mobile agents to client-server SNMP architectures. Bohoris et al [Bohoris et al 2000] discussed the use of mobile agents for performance management comparing it to CORBA and Java RMI-based solutions. Pagurek et al [Pagurek et al 2000] discussed reasons to integrate mobile agents with SNMP protocol and presented several ways of doing that. Schram et al [Schram et al 1998] presented a network modeling application based on mobile agents. The references above have shown that mobile agents' architecture has better scalability than client-server model and generate less management traffic in the network since management data volume is usually lower than the agents' code size. However, these solutions lack a greater support for access to heterogeneous management data formats. All of them are either limited to native access to management data, with specific code for each particular operating system, or to using a standard management base like MIB-II [McCloghrie and Rose 1991].

In recent years, we could also notice a relevant contribution from the IT industry, organized under the DMTF (Distributed Management Task Force), called WBEM (Web Based Enterprise Management) initiative [DMTF 1999a]. This initiative consists of the definition of a set of standards that provide a global and extensible information model to represent management information. The use of widely accepted internet standards to access this information is also a goal of this initiative. The information model proposed – CIM (Common Information Model) – is strongly based on object oriented concepts.

We have found few research initiatives that try to unify the benefits of these two approaches (WBEM and mobile agents). One proposed solution [Assis and Martins 2001] is based on a middleware that maps information obtained from CIM to an XML (Extensible Markup Language) representation. It grouped managed devices into *Agencies* and the communication between two agencies uses mobile agents. Notice that mobile agents are responsible only for transporting data between two agencies. They do not perform any analysis or take any decision based on the collected data.

Another contribution [Job and Simões 2002] presents a review of basic WBEM standards and describes experiments using several operating systems platforms. The implementations were based on a client-server architecture and confirmed the known limitations of this architecture.

In this work, we evaluate if the known benefits from mobile agents and WBEM approaches are still present when we unify the two solutions. While the use of mobile agents in this setting lacks support for heterogeneous management information models, the WBEM approach supplies a highly standardized, expressive and extensible information model. On the other hand, WBEM is based on a client-server model, supported by HTTP messages exchanged between clients and servers. Mobile agents supply an alternative to this architectural model reducing systems management overhead.

We expect that the two approaches are complementary. In this article we present a solution for systems management that combines their benefits. To meet this goal, we developed a framework to support intelligent mobile agents capable of obtaining information natively from CIM. To test our solution we have used two systems management prototypes.

In the next section we briefly review the basic WBEM concepts and emphasize the relevant aspects to this work. We then present an overview of the use of intelligent mobile agents for systems management. In section 4, we describe our solution that combines the benefits of these two approaches, and finally we present our results and conclusions.

2. Web-Based Enterprise Management (WBEM)

Web-Based Enterprise Management (WBEM) [DMTF 1999a] is an initiative from industry, organized by the DMTF (Distributed Management Task Force), to provide a set of standards for Systems Management (SM). These standards focus on supporting heterogeneous systems and are based on well known web standards such as HTTP (HyperText Transfer Protocol) and XML (eXtensible Markup Language).

WBEM is composed of three main standards:

- CIM (Common Information Model);
- xmlCIM;
- CIM Operations over HTTP.

CIM [DMTF 1999b] is a data model to represent management information from managed objects. XmlCIM [DMTF 2003b] is a standard for parsing CIM information to an XML representation. The third component, CIM Operations over HTTP [DMTF 2003a], is a transport model for XML-coded CIM information. Although WBEM does not restrict itself to the use of client-server model in management systems, CIM Operations over HTTP facilitate the use of this architectural model with WBEM standards.

In the next subsections we describe the relevant WBEM components for this work: CIM and CIMOM (CIM Object Manager) infrastructure.

2.1. Common Information Model (CIM)

Information used to perform actions is organized in such a way that distinct groups of people can use this information efficiently. To achieve this organization, it is necessary to have a model to represent the relevant details for each group of people. This approach is named *information model*. CIM is an information model that captures some common aspects needed for managing complex computer systems.

CIM is organized in three levels:

- *Core Model* – an information model that captures common notions applicable to all management sub-areas.
- *Common Model* – a set of information models that represents common details relevant to specific management sub-areas. This model groups information and

details for each sub-area, but it is not related to any specific technology or implementation. The sub-areas covered by the Common Model are: systems, applications, networks and devices. The *Common Model* and the *Core Model* compose the *CIM Schema*.

- *Extension Schemas* – represent technology-specific extensions from the *Common Model*. These extensions are specific to a particular environment, like an operating system or a hardware architecture.

CIM is an object-oriented information model and its basic concepts are defined by DMTF in the CIM *Meta Schema* [DMTF 1999b]. All basic concepts, like schemas, classes, properties, methods, triggers, indications, associations, references and qualifiers are defined in this Meta Schema.

CIM specification provides interoperability between CIM and previous well-known information models for network management, like MIBs [McCloghrie and Rose 1991]. In this way, CIM is concerned with extending, not replacing, previous models.

2.2 CIMOM Infrastructure

The WBEM architecture comprises managed systems and management systems. Managed Systems are hosts that support systems, which are managed by a management application. Management Systems are hosts where management applications execute.

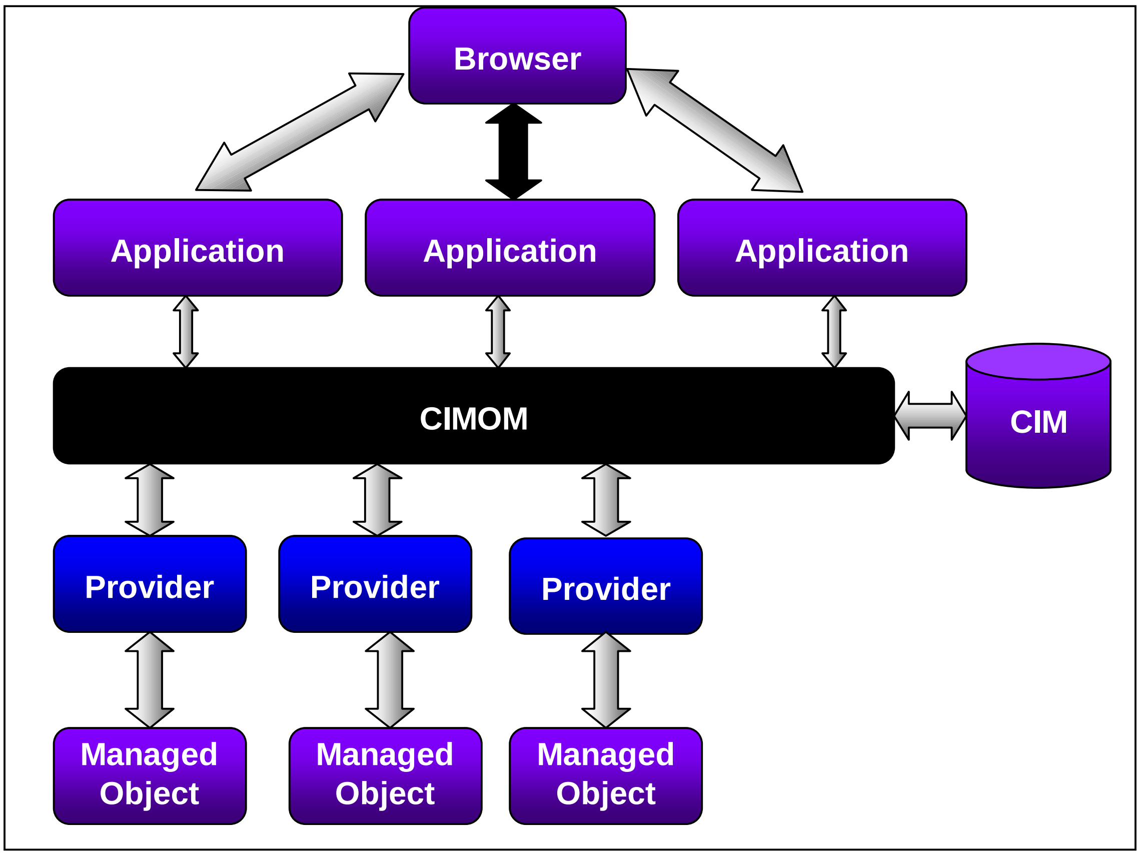

Figure 1 – The CIMOM infrastructure

All systems in a network must have a CIM object manager – CIMOM – that controls all CIM objects in a system. CIMOM is responsible for receiving local and remote requests about objects, their properties and methods and for sending answers to these requests. Figure 1 shows the CIMOM infrastructure.

CIMOM must get management information from several managed objects under its home system. These objects are of different kinds and have information represented in different formats. To accomplish this task and support heterogeneous managed objects, CIMOM uses auxiliary processes called *Providers*. *Providers* are processes capable of getting information directly from managed objects using native interfaces. *Providers* translate this information to the CIM format and send it to CIMOM.

There are two kinds of management information: static and dynamic. Static information is collected by CIMOM from a provider and stored in the CIM database. This kind of information is not real-time and does not require frequent updates. Dynamic information is real-time and is collected by CIMOM from a provider when it is requested by an application. CIMOM does not store dynamic data in the CIM database. This kind of data is always collected directly from a provider.

If a new managed object has to be managed by a CIMOM it is necessary to create a provider for this object. This process of implementing providers for all managed objects of a system is called system instrumentation. Providers are the tools used by CIMOM to manage heterogeneous objects.

CIM aims to become a global information model for systems management. When compared to other information models, CIM is considered the most expressive and extensible model for systems management [Vergara et al 2003].

3. Mobile Agents Approach for Systems Management

The amount of management information necessary to perform management tasks increases with the complexity and distribution level of current systems. This increment results in a growing overhead in the network traffic in traditional client-server management systems.

In recent years, several researchers have been working with new architectural approaches for systems management using Intelligent Mobile Agents. In this approach, a software agent is a computational entity which acts on behalf of others, is autonomous, proactive, and reactive, and exhibits capabilities to learn, cooperate and move [Bieszczad et al 1998]. This extends the traditional concept because the agent is not only reactive, answering to information requests, but is also proactive. The agent is not only capable of analyzing the information collected, but also takes decisions and performs actions independently from a central server. This behavior is possible due to development of Artificial Intelligence (AI). An agent which exhibits the capability of moving to different hosts is called a Mobile Agent.

In a Mobile Agent approach, the agent moves to a *host*, obtains information about local managed objects, processes this information, performs some action (if necessary), and moves to another *host*, repeating this sequence. This is the general behavior in a mobile agents approach. If we use a pure mobile agent’s solution, there is no central server and all processing is distributed among the several devices of the network.

To support mobile agents, each host must have a middleware layer that we will call *mobility framework*. This framework supports the process of migration between two hosts. This is called the agent's navigation model. It also supports the agent's life cycle model, computational model, security model and communication model [Bieszczad et al 1998]. These models, supported by a mobility framework, allow the agents to cooperate, move and use multithreading facilities.

An intelligent mobile agent has some reasoning capability that is usually implemented using forward or backward chaining engines [Russel and Norvig 1995]. This reasoning capability is essential to provide autonomous behavior to an intelligent agent.

4. Unifying the Two Approaches

In this section we describe our approach to unify the previous techniques in an integrated solution for systems management. Subsection 4.1 provides an overview of the proposed solution. Implementation decisions are described in subsection 4.2. Subsection 4.3 presents the tests and results used for validation.

4.1. Proposed Solution

The proposed solution is divided in two main modules. The first module is called *Maf* (Mobile Agent Framework). This module supports intelligent mobile agents and their main functions, like reasoning, mobility and communication. This module provides a generic support for intelligent mobile agents that may be applied in several domains of knowledge.

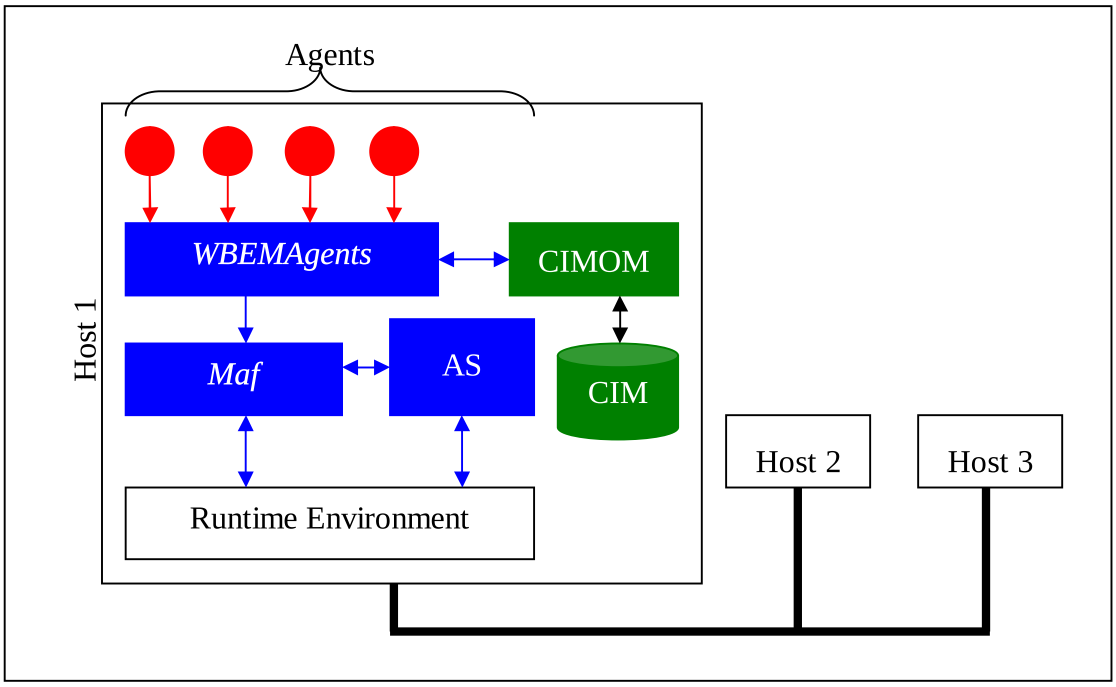

Figure 2 - Proposed solution architecture

The second module is called *WBEMAgents* and provides support for specific systems management activities. The main capability supported by this module is the native access to CIM. *WBEMAgents* defines a default navigation model for systems management agents, but provides freedom for developers to override this default model and decide what information should be collected from CIM and the agent life cycle. This way, this module is highly flexible and reusable in many systems management sub-areas.

An auxiliary component is the *Agents System* (AS). The AS is a piece of software that must be present in all hosts that are eligible to be visited by an agent. AS complements some of the *Maf*'s functions, such as migration, communication and life-cycle model support.

All these components are supported by a runtime environment that may be an operating system or a virtual machine. Figure 2 shows the proposed solution's architecture.

The motivation for encapsulating WBEM and mobile agents' support in a framework is the possibility to provide in a single programming interface access to CIM objects and mobile agents functionalities. With this framework, a system manager can create agents using high-level languages, like the production rules illustrated in figures 3 and 5. That way, the agents may be programmed by a manager with little knowledge of programming languages.

4.2. Implementation

The runtime environment used for this work was the *Microsoft .NET Framework* [Microsoft 2002], which complies with the *Common Language Infrastructure* (CLI) standard [ECMA 2001a, ISO 2003a]. Therefore, this implementation may be executed on any platform that has a CLI-compliant virtual machine. In choosing to support this runtime environment we are also offering the possibility of researchers on intelligent mobile agents to use this platform. Almost all recent work in this area has been based on the Java programming language and its runtime environment, the *Java Virtual Machine*. The most important reason for the environment choice was the stability of the implementation of CIM and CIMOM support in this platform. In previous work [Job and Simões 2002], we have noticed that the WMI (*Windows Management Instrumentation*) is one of the most stable and complete implementation for CIM and CIMOM. Using this environment, we did not need to implement new providers or API's to access CIM, which would be out of the scope of this work.

The choice of a runtime environment was necessary only for validation purpose. The architecture we describe here can be implemented over any runtime environment with no restrictions.

We have used the C# programming language [ECMA 2001b, ISO 2003b] because it has a syntax and semantic compatible with the object oriented paradigm and has easy access to all resources provided by the chosen runtime environment.

4.3. Validating the Solution

To validate the solution presented in previous subsections, we implemented two prototypes that execute some basic systems management tasks. For this implementation we used hosts with Microsoft Windows 2000 operating system and Microsoft .NET Framework runtime environment. The hosts were interconnected by a 100 Mbps local area network. All the implementations were based on open standards like CIM and CLI.

The first prototype was a system to verify logical disks availability. This system was developed as an agent that moves through the hosts in the network verifying if the amount of space available in each logical disk is less than 10% of total disk size. When this condition is found for any disk, the agent sends a message to a predefined management station.

This agent is implemented with a little knowledge base that is listed in figure 3. Every agent based on the *WBEMAgents* module has a rule base variable called *WBEMclass*. This variable's value is set by *WBEMAgents* when objects are read from CIM. *WBEMclass* always has a value indicating the type of CIM object that has already been read by the agent. For our disk space verifier agent, only CIM instances of type *Win32_LogicalDisk* are relevant. When the agent reads an instance of this type, it checks if it refers to a local hard disk and, if so, sets the variable *localDisk* to *Yes*.

When the agent has an instance representing a local hard disk and its total size is greater than zero, it calls an effector called *calcPerc*. This effector is responsible for calculating the percentage of available space in the disk. If this is less than 0.1, then the effector *sendMsg* is called to send a message to a predefined management station.

```
Rule Calc: IF WBEMclass = Win32_LogicalDisk
               AND localDisk = Yes
               AND size > 0
             THEN effector(calcPerc)
Rule Local:IF WBEMclass = Win32_LogicalDisk
               AND driveType = 3
             THEN localDisk = Yes
Rule diskFull: IF localDisk = Yes
                   AND percent < 0.1
                 THEN effector(sendMsg)
Rule stop: IF done = Yes
             THEN effector(stopAgent)
```

Figure 3 – The knowledge base for a disk space verifier agent

The variable *done* is set to *Yes* by *WBEMAgents* when all programmed hosts have been visited by the agent. When this variable is set to *Yes* the effector *stopAgent* is called to kill the agent and finish its processing. If this variable is not set to *Yes*, *WBEMAgents* will try to move to the next host to continue processing.

On each visited host the agent will write a text file named *Win32_LogicalDisk.WBEM*, with all properties of all instances of this type and their respective values. This file also registers the date and time when the data was collected.

The purpose of this file is to be available to other agents in the future, who may want to compare the real-time data with the last collected data.

The *effector* is an important element in this knowledge base. We have, in figure 4, three effectors. By using effectors an agent can act over its knowledge base and over the environment. For example, the effector *sendMsg* could be replaced by another effector that sends e-mail messages or another effector that sends a message to other agents that would try to solve the disk space problem. We could imagine an effector that tries to delete temporary files to increase the percent of free disk space, for example. The effectors add flexibility to the solution.

In all tests executed, the agent achieved 100% success in identifying logical disks with less than 10% of free disk space. This agent doesn't collect information during its travel in the network. The information read from CIM is processed locally on each host. Therefore the agent's size has no increment during the task execution. Actually, this size is decreased because the pending host list is being reduced on each agent migration. The average agent size, in this example, is 6.28 Kbytes.

On each agent migration, 6.28 Kbytes of traffic is generated in the network. To verify nine hosts in the network, the total traffic generated was 50.23 Kbytes. The average time needed to complete the task was 18.16 seconds. Therefore we have an average throughput of 2.77 KB/s.

Using another solution based on a stationary agent that collects disk information using the client-server paradigm, we generated 140.4 Kbytes of traffic to verify nine hosts. Therefore we have three times more traffic in the client-server solution than in the mobile agents' one. This difference may be explained because in the client-server model we need to read all instances of type *Win32_LogicalDisk* from each host over the network and all this data is processed in a central point. So we have an average traffic of 15.6 Kbytes per host processed in the client-server solution. In the mobile agents solution the average traffic is obtained by the formula *(n-1) x 6.23*, where *n* is the number of hosts visited. Figure 4 compares the traffic generated in the network by these two solutions.

The second prototype is a Report solution. One of the most common tasks executed by a systems manager is to get updated and concise information about the managed systems. This sub-area is called Asset Management. We developed a simple Report System based on mobile agents. We also developed a version using the client-server model for comparison purposes.

Our prototype is an agents system capable of collecting common hardware information from each host visited and, in the end of collection process, it generates a report. In this example the information collected was on the type of processor, memory and disks.

To achieve this goal, the agent gets property values from CIM objects of types *Win32_Processor*, *Win32_LogicalMemoryConfiguration* and *Win32_LogicalDisk*. We selected some relevant information from these CIM types to build our asset report.

As in the first prototype, all the reasoning is modeled through a knowledge base presented in figure 5. This rule base is composed by four rules. The first three rules are responsible for starting effectors that collect processor, memory and disk information, respectively. The last rule verifies if the task is done and starts an effector responsible for generating the report, sending a notify message and stopping the agent.

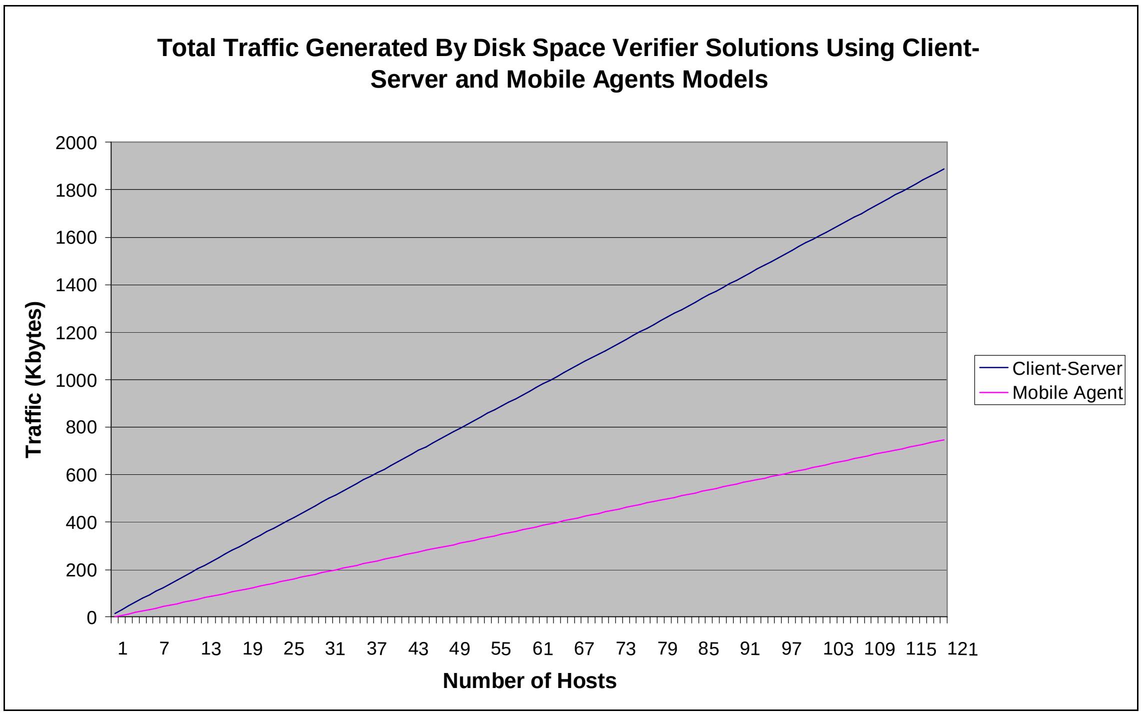

Figure 4 – Total traffic generated by disk space verifier solutions using client-server and mobile agents models

```
Rule addCPU: IF done = No
                  AND WBEMclass = Win32_Processor
                THEN effector(collectInfo, "CPU")
Rule addMemory: IF done = No
                 AND WBEMclass = Win32_LogicalMemoryConfiguration
                  THEN efffector(collectInfo, "Memory")
Rule addDisk: IF done = No
                    AND WBEMclass = Win32_LogicalDisk
                  THEN effector(collectInfo, "Disk")
Rule end: IF done = Yes
             THEN effector(genReport)
```

Figure 5 – Knowledge base for a report generator agent

In the first set of tests we used a single agent system to generate the report. In this experiment, we varied the number of hosts to be visited. For each number of hosts we executed twenty times the same task, so that we could obtain an average of the system behavior.

In a second set of tests, we used a system composed of two agents. The agents divide the number of hosts to visit, so that one agent visits half of the total number of hosts and the other agent visits the other half. In a third set of tests, we developed a client-server solution to generate the same report. In our fourth set of tests, we used an implementation with 5 agents to solve the same problem. The time needed to complete the task in these four versions, varying the number of hosts, can be viewed in figure 6.

We can observe that the client-server solution is slightly better than the solution with one agent. This can be explained by the fact that the agent has a growing code size. Therefore, when the number of hosts increases, the agent's size also increases, and the amount of time needed to move this code over the network becomes grows. The four solutions have an approximately linear growth. But we should notice that the solution with two agents has a slower growth rate than the client-server one. The slowest growth rate and the best performance among the solutions may be observed in the implementation with 5 agents. This shows that increasing the number of cooperating agents we obtain a reasonable scalability growth. However, if we use too many agents to accomplish one task, we may notice a growing throughput of network traffic due the frequent agent migrations in short time intervals. The cost of inter-agent communication, coordination and agent management may become overwhelming if we continue to increase the number of agents.

When we measured the generated traffic in the network, this second prototype has a different behavior from the first prototype. This agent collects information during its travel through the network. This way, the agent size increases on each migration and the generated traffic has an exponential growth trend as we increase the number of hosts to visit.

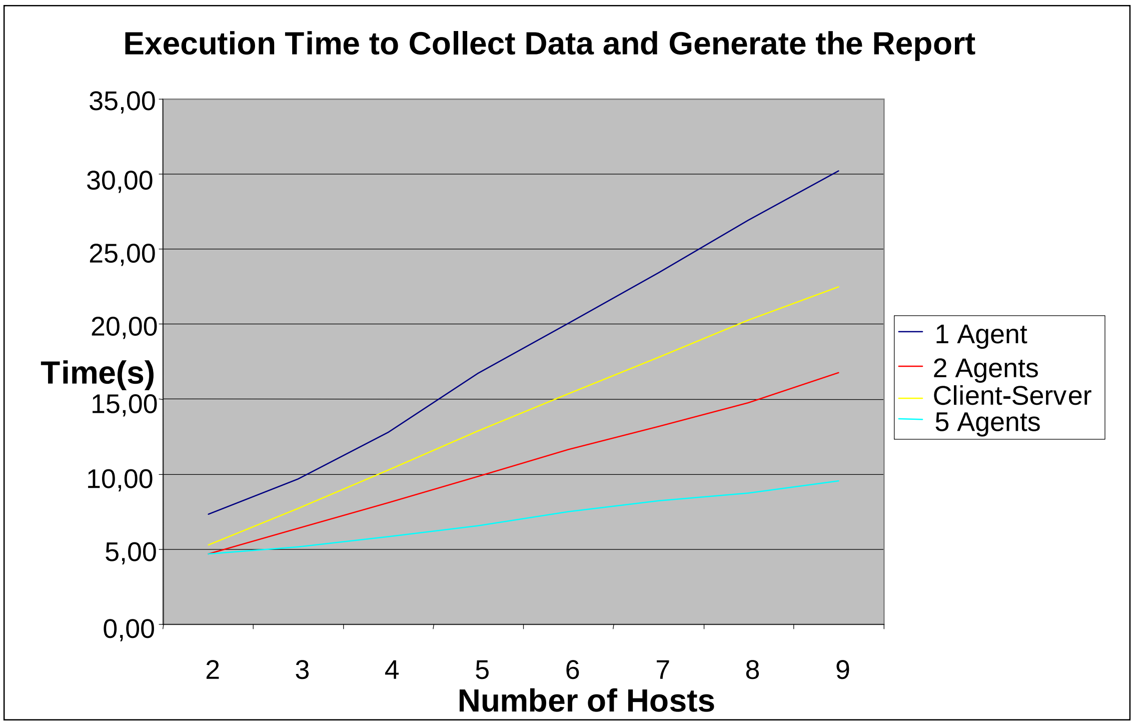

Figure 6 – Execution time for the report generator systems

In our example, the agent initial size is 7.98 Kbytes and it increases by 1.58 Kbytes on each migration. As we can see in figure 7, the client-server solution is better than 1-agent solution for a number of hosts greater than 21. And it is better than the 2-

agent solutions for a number of hosts greater than 61. But the 5-agent solution is better than the client-server one for the number of hosts measured in the tests. This shows that the increase in scalability with the use of more agents is confirmed in terms of traffic volume. There is a trend to confirm the results presented by Rubinstein et al [Rubinstein et al 2000], where the better performance of the mobile agents' solution is limited between two boundaries in terms of the number of managed elements.

The last measurement collected in this second prototype was processor time consumed by the solutions. In the agents' solutions we obtained an average overhead of 0.98% of total processor time on each host. In the client-server solution we had a 6.14% overhead of total processor time on the management station, where all the processing was done. In this item, the most relevant information is not the total time consumed by the solutions, but the distribution characteristic of this processing. More complex management tasks would require a very powerful processor on the management station in client-server solutions, but in mobile agents' solutions this processing is distributed over all the hosts in the network, reducing the need for powerful centralized processors.

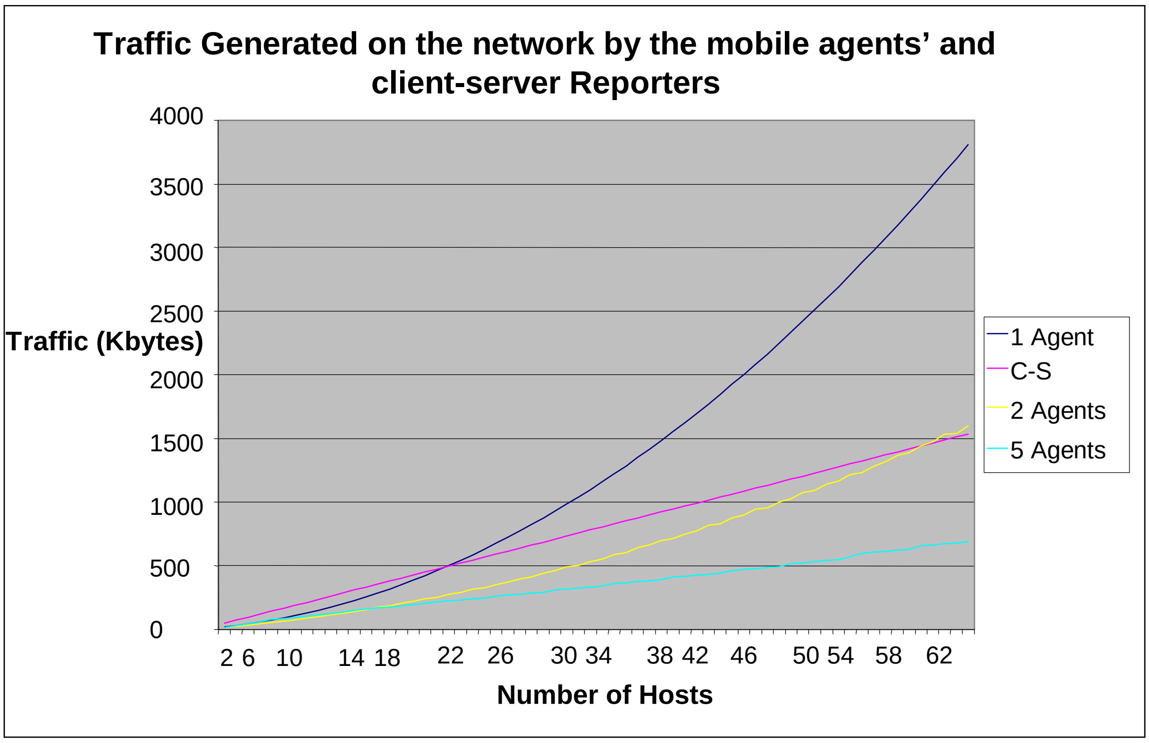

Figure 7 – Traffic generated on the network by report generator systems

5. Conclusions

In this work, we have evaluated the use of intelligent mobile agents with standards provided by WBEM initiative. We have shown that benefits like better scalability and less management traffic are still present when mobile agents are used with CIM, under certain conditions. We have noticed that by tuning the number of agents involved in a management task we may optimize the total time necessary to perform that task and reduce network traffic. But the increase in the number of agents can not be unlimited. An excessive number of agents may cause high throughput and a poor performance for the system.

When mobile agents read information from CIM, only the relevant properties from each CIM object are selected by the agents. In a common client-server implementation, the entire CIM object is read over the network and the management application selects the relevant properties only at the management station. This contributes to the reduction in management traffic when we unify mobile agents and CIM.

We must notice that previous work[Bohoris et al 2000] had different results in terms of response time. In those experiments mobile agents' had worse response times than CORBA-based and Java RMI-based client-server solutions. In terms of network traffic the results were similar to our results. We don't know if these differences are caused by the different application area (performance management) or by different client-server technologies that were used by the authors.

As future work we may enhance some characteristics like the communication model and add other WBEM standards like xmlCIM. *Maf* lacks a fault tolerance engine and an enhanced security system. We may also validate *Maf* in other operating systems platforms that support CLI compliant runtime environments, like Linux, FreeBSD and Mac OS X. Finally we can test the efficiency of this solution in other systems management sub-areas, like performance management or configuration management.

In this work, we presented important contributions towards the development of highly scalable systems management tools that support the distributed applications scenario and the diverse heterogeneous systems that are often present in corporate networks.

References

[Assis and Martins 2001] Assis, P. and Martins, J. A. "XML Based Resource Management: a CIMOM Approach". In: Proceedings of Second Latin American Network Operation and Management Symposium (LANOMS'01). Belo Horizonte : Federal University of Minas Gerais, p. 307-318, 2001.

[Bieszczad et al 1998] Bieszczad, A. Pagurek, B.; White, T. "Mobile Agents for Network Management". IEEE Communications Surveys. IEEE, v. 1, n. 1, 1998.

[Bohoris et al 2000] Bohoris, C; Pavlou, G.; Cruickshank, H. "Using Mobile Agents for Network Performance Management". In: Proceedings of IEEE/IFIP Network Operations and Management Symposium (NOMS 2000). IEEE, 2000, p. 637-652.

[Coulouris et al 2001] Coulouris, G., Dollimore, J. and Kindberg, T. Distributed Systems: Concepts and Design. USA : Addison Wesley, 3 ed, 2001.

[DMTF 1999a] Distributed Management Task Force, Inc. Web-Based Enterprise Management (WBEM) Initiative. DMTF, 1999.

[DMTF 1999b] Distributed Management Task Force, Inc. Common Information Model (CIM) Specification. DMTF, 1999.

[DMTF 2003a] Distributed Management Task Force, Inc. Specification for CIM Operations Over HTTP. DMTF, 2003.

[DMTF 2003b] Distributed Management Task Force, Inc. Specification for the Representation of CIM in XML. DMTF, 2003.

[ECMA 2001a] ECMA International. Common Language Infrastructure (CLI). Standard ECMA-335, 2001.

[ECMA 2001b] ECMA International. C# Language Specification. Standard ECMA-334, 2001.

[ISO 2003a] International Organization for Standardization. Common Language Infrastructure. ISO/IEC-23271, 2003.

[ISO 2003b] International Organization for Standardization. C# Language Specification. ISO/IEC-23270, 2003.

[Job and Simões 2002] Job, J. C. C. and Simões, M. A. C. "Gerenciamento de Sistemas Baseado no Padrão WBEM". In: Scientific Initiation Electronic Magazine. Computation Brazilian Society, v. II, n. IV, 2002.

[McCloghrie and Rose 1991] McCloghrie, K. and Rose, M. Management Information Base for Network Management of TCP/IP-based internets: MIB II**.** RFC 1213, IETF, 1991.

[Microsoft 2002] Microsoft Corporation. Microsoft .NET Framework: Product Overview. 2002.

[Pagurek et al 2000] Pagurek, B., Wang, Y., White, T. "Integration of Mobile Agents with SNMP: Why and How ?" In: Proceedings of IEEE/IFIP Network Operations and Management Symposium (NOMS 2000). IEEE, 2000, p. 609-622.

[Rubinstein et al 2000] Rubinstein, M. G., Duarte, O. C. M. B. and Pujolle, G. "Evaluating the Network Performance Management Based on Mobile Agents". In: Proceedings of the Second International Workshop on Mobile Agents for Telecommunication Applications (MATA'2000), Paris : Springer-Verlag, p. 95-102, 2000.

[Russel and Norvig 1995] Russel, S. J. and Norvig, P. Artificial Intelligence: A Modern Approach. New Jersey : Prentice-Hall, 1995.

[Schram et al 1998] Schram, C., Bieszczad, A. and Pagurek, B. "Application-Oriented Networking Modeling with Mobile Agents". In: Proceedings of IEEE/IFIP Network Operations and Management Symposium (NOMS'98). New Orleans : IEEE, 1998.

[Vergara et al 2003] Vergara, J. E. L. de, Villagrá, V. A., Asensio, Juan I. and Berrocal, J. "Ontologies: Giving Semantics to Network Management Models". In: IEEE Network special issue on Network Management. IEEE, 2003.