A scalable SNMP-based distributed monitoring system for heterogeneous network computing

Rajesh Subramanyan, José Miguel-Alonso and José A.B Fortes
School of Electrical & Computer Engineering, Purdue University, W. Lafayette, IN-47907.
{*subraman,miguel,fortes*}*@ecn.purdue.edu*

Traditional centralized monitoring systems do not scale to present-day large, complex, network-computing systems. Based on recent SNMP standards for distributed management, this paper addresses the scalability problem through distribution of monitoring tasks, applicable for tools such as SIMONE (SNMP-based monitoring prototype implemented by the authors).

Distribution is achieved by introducing one or more levels of a dual entity called the Intermediate Level Manager (ILM) between a manager and the agents. The ILM accepts monitoring tasks described in the form of scripts and delegated by the next higher entity. The solution is flexible and integratable into a SNMP tool without altering other system components.

A testbed of up to 1024 monitoring elements is used to assess scalability. Noticeable improvements in the round trip delay (from seconds to less than tenth of a second) were observed when more than 200 monitoring elements are present and as few as 2 ILM's are used.

1 Introduction

Wide-area network-computing systems (NCS) are currently seen as a promising way of efficiently using the tremendous computational power available in computers connected to the Internet. A monitoring mechanism is an essential part of these systems. The information it provides is used for resource management, scheduling applications, trouble-shooting, providing performance information to network-aware applications, etc.

Monitoring a NCS goes beyond a mechanism to just gather information. From the users point of view, the purpose of monitoring encompasses what needs to be measured, at what rate are measurements required, what is the acceptable delay, and what are the acceptable overheads. In this high-performance, distributed computing scenario, the performance of the monitoring tool itself becomes an issue of concern along with the performance of the system it seeks to measure. This paper proposes mechanisms and a strategy for efficiently distributing monitoring tasks across an NCS.

Heterogeneous components in a NCS can be monitored by using standardized network monitoring and management protocols such as SNMP [1]. Most computers currently implement SNMP daemons (agents) that provide information about their status. Using these daemons and protocols helps dealing with interoperability and reduces implementation costs.

Most packages based on SNMP are centralized: a single manager collects information from many agents. The centralized architecture has important advantages, e.g. simplicity and that it fits well many monitoring applications (see Figure 1): those that have little need for distributed administrative control, do not require frequent polling or high frequency computation, have high-throughput network connectivity between manager and agents and pass around a small amount of information [2]. An important problem of this approach is scalability. An additional concern is fault tolerance (what happens if the manager fails?). This paper focuses on the scalability of the monitoring functions and briefly discusses fault-tolerance issues, a subject of ongoing work, in Section 6.

In a centralized monitoring system, the manager receives a large volume of data through its network interface. This data also traverses several links to reach the manager, thus increasing the packet traffic per network link. The computations performed with the received data, although simple, are numerous, resulting in high CPU usage of the manager. Thus, the manager and the network links close to it can easily become a bottleneck. Distribution reduces overheads and thereby improves scalability; it is also a step towards improving reliability. However, distribution also brings out new challenges: lack of global control, synchronization, reduced standardization and increased complexity.

Centralization and distribution need not be seen as

0-7803-9802-5/2000/$10.00 (c) 2000 IEEE.

opposing solutions. Rather, they are two end points on a scale with many intermediate points. Depending on the target system and the required performance, the choice of architecture may vary on this scale. Ideally, the architecture should dynamically adapt to the user's requirements.

In [3], we describe the design and evaluation of SIMONE, a modular, SNMP-based monitoring system targeting NCS. Parameters measured by SIMONE are under the broad areas of general monitoring information (e.g. host description), application monitoring (CPU time, memory used by a process), machine monitoring (CPU load, average page fault rate), network interface and link information (traffic in/out of host, delay between hosts) and monitoring overhead measurements. See [3] for a complete list of currently implemented parameters. SIMONE has a user interface component that allows control of the different operations of the monitor, and a user-friendly GUI to present results in different forms such as result summaries, tables and graphs (delayed and real-time). Work is in progress on a library-based API that will allow other applications to interface with SIMONE.

Other monitoring tools have been developed targeting NCS from different perspectives (like NWS [4], Netlogger [5], Remos [4], Gloperf [6] etc., see Section 5). Although some of these take advantage of utilizing additional SNMP information, they differ from SIMONE in the sense that their core technology is not SNMP.

SIMONE currently operates as a centralized system, but its design was done with distribution in mind. In this paper we propose enhancements to distribute SIMONE's monitoring tasks. The key idea is to move these to the data they process, as opposed to the traditional approach of moving the data to the task. Moving tasks is more efficient if the amount of data that needs to be transferred is large or the transfer medium is a potential source of unreliability. Our approach helps reducing communication costs, avoids a single point of failure and, most importantly, improves scalability. To realize this goal, a distinct entity called the Intermediate Level Manager (ILM) has been designed, implemented and integrated into SIMONE with minor changes to the manager and no alterations to the agent or any other module. The ILM is designed to be generic, permitting one or more layers to be interposed between manager and agents.

Monitoring tasks are delegated among a collection of ILMs' (Figure 2). An ILM act as a manager for a collection of agents, whereas, a collection of ILMs' act as agents to a higher-level ILM, or to a top-level manager (TLM). The ILM thus behaves as a dual entity. A tree is formed, with a TLM in top, several layers of ILMs' in the middle, followed by the agents at the bottom. Vertical communication is essential, while horizontal communication is an option that would allow ILMs' at the same level to cooperate. Our initial implementation does not support horizontal communication.

This distribution scheme is based on the most recent MIB proposals from the standardization committees, in particular, on the Script MIB [7] (see Section 5). The performance of this distributed approach is evaluated, although limited to study the effect of the number of nodes on the performance of the monitoring system.

The remainder of this paper is organized as follows. Details of the ILM architecture are given in Section 2. The experiments performed to evaluate the system are described in Section 3 and the results are analyzed in Section 4. Other distributed implementations are discussed under related work in Section 5. Conclusions and future work are discussed in Section 6.

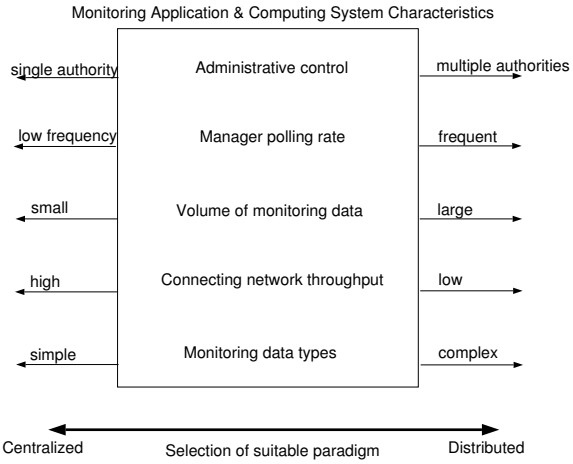

Figure 1: Selection of suitable paradigm based on computing system and monitoring application characteristics

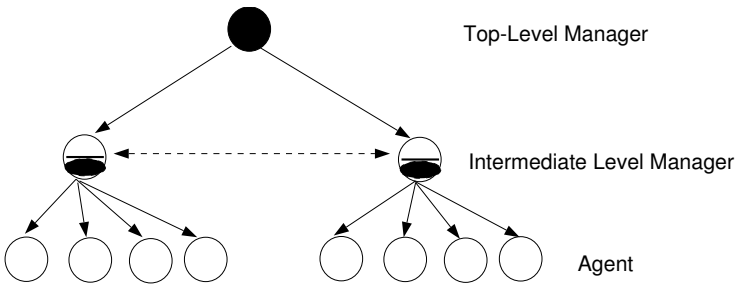

Figure 2: Distribution across vertical and horizontal axis. Vertical is referred to as delegation, and horizontal as cooperation. The shaded area denotes a manager or an entity performing the role of a manager. Non-shaded areas denote agents or agent role.

2 The Distributed SIMONE

In a centralized, SNMP-based monitoring system such as the current implementation of SIMONE [3], components are a collection of agents coordinated by a single manager. In terms of data collection, they collaborate in a client-server mode: the manager requests and obtains monitoring data, thus performing the role of a client, while the agents listen and respond to requests, thus acting like servers (see Figure 3). The role of the manager is however more complex, because it has to control the entire system and additionally process and perform computations with the received data. The various operations performed by the monitoring system or the manager in the centralized system are referred to as *monitoring tasks*. Manager communicates with agents via SNMP set and get protocol primitives. A [sessionpid set variable value] primitive allows the manager to set the value of a MIB variable of the target agent (specified by sessionpid), while a [sessionpid get variable] primitive retrieves the value of a MIB variable from the agent (described by sessionpid). Agents respond to requests from the manager by assembling the data in a SNMP packet and tagging the address of the manager in its header.

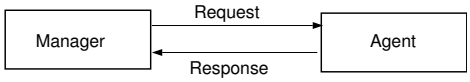

Figure 3: Centralized client-server model

In the case of the distributed monitoring system described in this paper, the role of manager and agents as defined by SNMP is preserved. However, the monitoring tasks are distributed along a collection of ILM's that are located between the manager and the agents. Figure 4 shows a simple structural model of the ILM. It is a dual entity, whereby it plays the role of both a manager as well as an agent. From the point of view of the top-level manager, the ILM is just an (enhanced) agent. From the point of view of the agents, an ILM is just a regular manager. Introducing this new layer in the monitoring architecture does not require any change at all in the design of the agents. The manager requires minimum modifications in order to take advantage of this collection of entities that help reduce its workload. Several layers of ILM's can be introduced without any changes to the basic code of the ILM in order to form a hierarchical structure. The partitioning of nodes assigned to ILM's can be done based on network domain, or by partitioning the task (specially for large tasks), or by administrative domains. Our solution is generic in the sense it can handle all the above cases. Currently, the user specifies the allocation of nodes to the various ILM's. In the same way that SIMONE closely follows the SNMP standards, the proposed distribution mechanism is based on the Script MIB [7], one of the latest proposals of the DISMAN (Distributed Management Group) of IETF.

The ILM's are flexible entities, able to execute small programs (scripts) defined by the manager and at the manager's request. Mechanisms are needed to allow a manager to send a script to a ILM, and to trigger the actual execution of the script. The basic principle of this design is that no changes in SNMP should be introduced. For this reason, a script is sent to an ILM via a SNMP [set variable value] command, where the data sent (the value) is the script or a reference to it. The ILM binds the variable with the script in its MIB. The triggering mechanism is based in the [sessionpid get variable] command: when the manager request the value of a variable that is bound to a script, the ILM does not return the value of that variable. Instead, it passes control to its "manager half", performs the appropriate tasks (that could include requesting data to a collection of agents, retrieving it and processing it) and, when the script has been completed, and a result has been obtained, the corresponding value is sent to the manager (see Figure 5).

Scripts can be written in any language, provided that the ILM understands it. Typical choices are shell languages, Java, Perl, Tcl or even proprietary languages specifically designed for the purpose of manipulating MIB objects. The design of the system is independent of the language of choice. In our prototype system, we use Tcl.

Regarding the mechanism to distribute scripts, there are two basic choices, already mentioned: sending the script as data in a set command (push model) or sending a reference to it, for the ILM to take care of retrieving it (pull model). The reference is a URL. An advantage of the pull model is that there is no limit in the size of the script, while in the case of the push model the maximum size is determined by the payload capacity of a UDP datagram. Currently we trigger a script from a set of pre-loaded scripts.

Summarizing, the essential functional capabilities required for ILM are:

- Dual operation capability
- Trigger mechanisms: capabilities for initiating, suspending, resuming and terminating management scripts.
- Communication: between the entity above the ILM and the ILM, and the ILM and the entity below the ILM. Communication provides capabilities for the following.

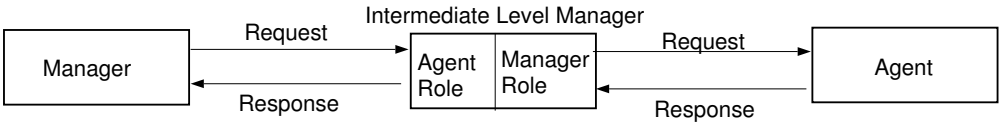

Figure 4: Decentralized model with the inclusion of the ILM

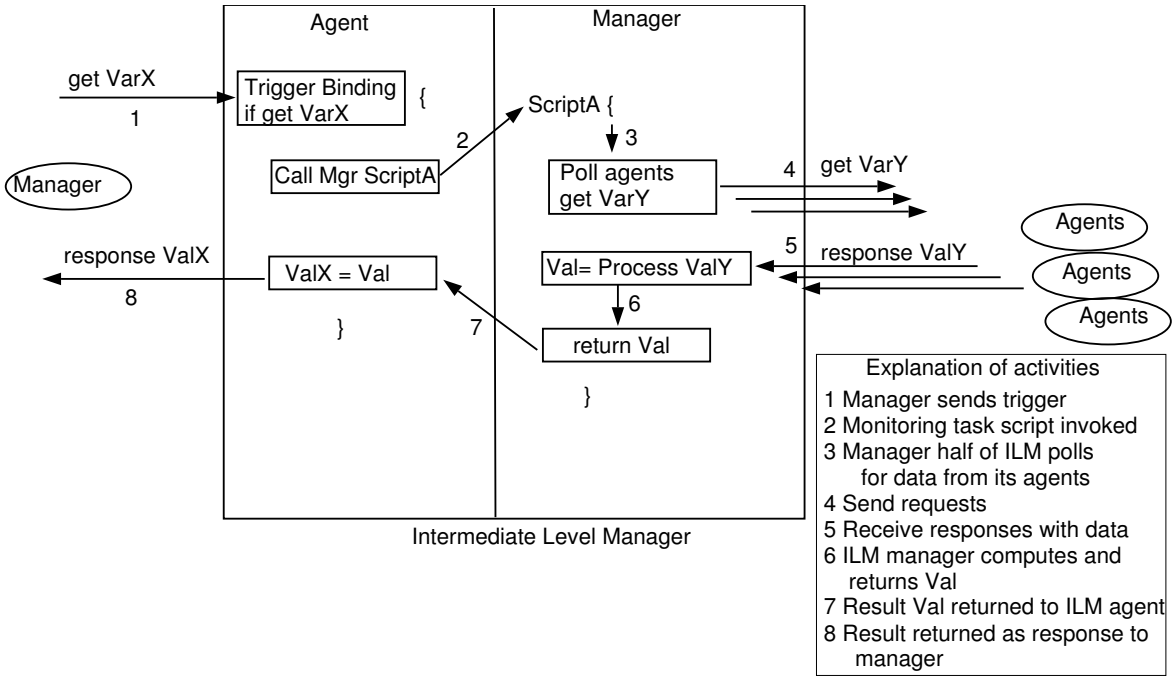

Figure 5: Simplified illustration of the functioning of the ILM

- to transfer management scripts to the ILM
- to transfer arguments for management scripts.
- to transfer the results produced by running management scripts.

3 Experiments

The experimental testbed consists of approximately 128 Sun workstations of different classes (Sun Ultra 30, Ultra 5, Ultra 1, Sparc Station 5, Sparc LX, etc.) on a local area network and running the Solaris 2.6 operating system. Each machine has multiple agents running, to obtain up to 1024 agents representing an equal number of monitoring elements, referred hereinafter as nodes. A single top level manager and all the intermediate-level managers run on a homogeneous set of machines (all Ultra 5's) selected from the same set. This prevents variations in the results due to differences in machine performance ratings. No such selection criterion is adopted for agents. Previous experimental studies on agent performance [3] have shown that load increase is less significant on the agent than on the manager under most operating conditions (except in cases where the agent is forced to be a bottleneck). Unless otherwise stated, all experiments performed are confined to a single layer of intermediate-level managers.

In order to evaluate and compare the performance of the centralized and distributed schemes, the monitoring system performs a simple task, in this case obtaining the least loaded node and the corresponding load. The time elapsed from the moment the top-level manager requests for information about the least loaded node and the moment this information is obtained is the *delay*. In addition to measuring delay, two other performance indicators, namely (i) *CPU overhead* and (ii) *communication overhead,* are measured.

Two sets of measurements are performed. In the first, the target system size is varied from 2 to 1024

nodes and the performance values for centralized and distributed cases are obtained and compared. The performance values show the degradation of performance with target system size, thereby indicating whether the monitoring system scales or not. In the second set, the system size is kept fixed and the number of intermediate-level manager's is varied. The performance values vary with the number of ILM's. This helps characterize the performance of the monitoring system for different distributed configurations as well as observe the "best" configuration for a given monitoring task.

In order to measure the CPU and communication overheads, the monitoring task used for the above experiments is repeated a number of times (about 20-25) at short intervals of time (2-3) seconds. The CPU load increase due to monitoring activity on the top-level manager machine and an ILM machine (both machines do not run any other user applications) are recorded for configurations with 256 nodes and ILM's ranging from 0 to 64. The number of monitoring packets/second on the network interface of these machines is also measured as the communication overhead.

To ensure the validity and accuracy of results some methods are employed. Among the large number of observations recorded, those that are affected significantly by network fluctuations are discarded. It is also ensured that for each reading all the nodes (agents) are functioning. Although the ILM functions even if an agent stops working, there is a slight variation in the total delay recorded. This is not permissible for performance tests since we need to make comparisons between two sets of readings observed under identical conditions.

4 Results and Interpretation

The delay in executing the monitoring task is measured for target sizes ranging between 2 and 1024 nodes for both the centralized as well as the distributed schemes. In the distributed scheme, the number of ILM's is varied between 2 and N/2 for a fixed target size of N elements.

A steep rise in delay in the centralized scheme is observed in Figure 6 for target sizes above 256. Although the value of permissible delay is application-dependent, assuming 0.5 s to be a reasonable value, the centralized scheme is suitable for target sizes less than or equal to 100-200 nodes. Beyond this size, the system is saturated, resulting in an exponential rise in delay. This suggests the need for distributed schemes for target sizes larger than 100-200 nodes.

Figure 7 compares the centralized scheme with different configurations of the distributed scheme.

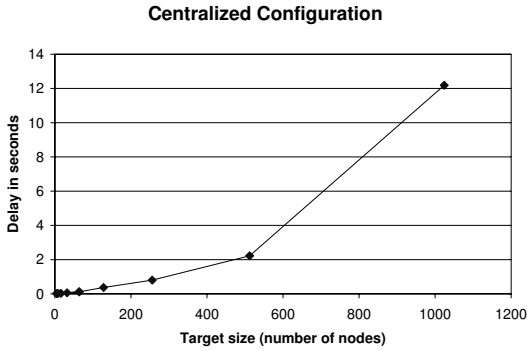

Figure 6: Delays in the centralized scheme for different target sizes

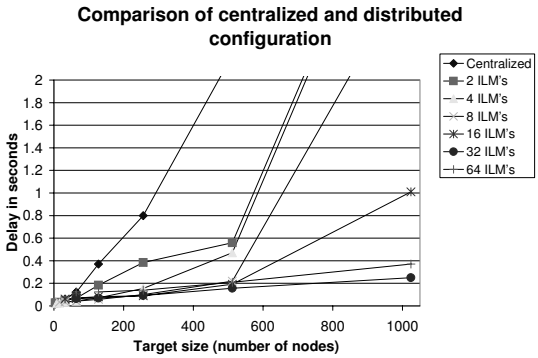

Figure 7: Delays in the distributed scheme for different target sizes

Weakly distributed configurations (few ILM's, e.g. 2, 4, etc.) have delays closer to the centralized scheme, while strongly distributed configurations (large number of ILM's) have shorter delays. A configuration with multiple levels of ILM's is useful only when a bottleneck persists due to the nature of the monitoring task and the target system. Otherwise, the use of multiple levels reduces the performance due to added communication delays at each level. For the selected monitoring task, and for the range of target size, a single level is sufficient. This point is illustrated for target size 256 by comparing the delay for one level of 64 ILM's, two levels of 7-56 ILM's (7 ILM's at one level, and 56 ILM's at the next lower level assigned equally, i.e. 8 ILM's for each higher level ILM), and three levels of 4-12-48 ILM's. The best results are obtained for a single level as shown in Table 1.

Figure 8 compares delays for a fixed target size. Among the distributed configurations it is observed that the optimal case is approximately 32 (i.e. in the range 16-64) ILM's for target size 512, approximately 16 ILM's for target size 256, and 8 ILM's for target size 128. It appears that the optimal number of ILM's is for a balanced configuration, i.e. $\sqrt{N}$. Table 2 compares the approximate optimal number of ILM's

Levels of ILM	Number of ILM's/level	Monitoring Delay in s
0 (centralized)	-	0.8
1	64	0.14
2	7-56	0.141
3	4-12-48	0.3

Table 1: Multiple levels of ILM for 256 nodes

Target Size	$\sqrt{N}$	Approximate number of ILM's for least delay (from Figure 8)	Range of of ILM's for least delay (from Figure 8)
512	22	32	16-64
256	16	16	8-32
128	11.5	8	4-16
64	8	8	4-8

Table 2: Optimal configuration of ILM's for given target size to minimize monitoring delay

obtained from Figure 8 for a target size N v.s. $\sqrt{N}$. For a configuration with too few ILM's (i.e. weakly distributed) the ILM's are the bottleneck (since an ILM has to manage a large number of agents) and for a configuration with too many ILM's, the manager is the bottleneck. The optimal number of ILM's may be below or above $\sqrt{N}$ depending on the characteristics of the monitoring task.

The CPU and communication overhead measurements for different distributed configurations (2-64 ILM's) for target size of 256 nodes provided the following results. The CPU load due to the given monitoring activity increases between 0 to 0.1 for the top-level manager and between 0 and 0.05 for an ILM, both being marginal increases. For the same setup, communication overheads due to monitoring is 30-50 packets/s for the top-level manager (50 packets/s measured while managing 64 ILM's). Communication overheads for an ILM shows a linear increase when the number of leaf nodes it supports increases. The overheads are between 4-65 packets/s for 8-128 nodes per ILM. The results indicate that for the target size and the monitoring task under consideration, both CPU and communication overheads values do not significantly change for different distribution configurations, and the values are far below saturation points.

5 Related Work

Numerous monitoring tools have been used for different purposes in heterogeneous distributed environments. A few tools have been recently designed for network computing environments. The NWS [4] provides dynamic resource performance forecasts in network computing environments. This information can be used to feed network-aware applications, and for resource scheduling. Monitoring is done via system calls and end-to-end tests, although recent extensions allows NWS to gather SNMP data. Globus' Globerf [6] tool works like NWS, and periodically schedules end-to-end tests to retrieve latency and bandwidth information. In both cases, scalability is improved by using a hierarchy of groups. As a result the measurement is more complex, but the benefit is that the number of tests is reduced. Netlogger [5] is a trace-based system used mainly for diagnosis. The scalabilty problem is tackled by starting and stopping logging processes. Remos [8] combines different monitoring techniques such as SNMP and end-to-end tests with the focus on providing information for network-aware applications. The monitoring architecture is hierarchical, i.e. a data collector is in charge of a subnet and an upper-layer collector consolidates information from several subnets. Scalability is thus achieved as a side-effect of this distribution. All of these tools use non-standard communication protocols. Use of SNMP is seen as an enhancement, and not as the core technology as is the case with SIMONE.

The Grid Performance Working Group of the Grid Forum is in the process of defining a monitoring service architecture for the Grid [9]. The authors intend to make SIMONE compliant with these anticipated framework specifications, so that SIMONE can interact with other Grid systems.

Another line of related work comes from the SNMP community and is discussed in the remainder of this section. The idea of decentralizing SNMP monitoring tasks is as old as the protocol itself. Although Remote Monitoring Management Information Base [10] (RMON MIB), offered the first and the simplest form

of delegation (agent collates without manager intervention thus reducing overheads), it has not been actively pursued as a scalable solution. SNMPv2 proposed enhancements to retrieve large blocks of data, and provide *cooperative* communication to support a weakly distributed hierarchical model. SNMPv2, however failed to be accepted due to practical problems and most of the proposed enhancements were discarded [11]. Weakly distributed hierarchical models only partially address the problem of scalability and lack flexibility and robustness. To address this, strongly distributed hierarchical models emerged in the mid nineties.

The MbD [12] model proposed first in 1991, was a milestone which triggered several research efforts, prototype implementations and even helped the standardization efforts by IETF and ISO, who much later integrated this model into their standards. In the MbD approach, management functions are delegated to remote devices, thus reducing communication costs and increasing scalability. A Remote Delegation Protocol (RDP) is used to distribute tasks. The work in this paper uses the idea of delegation. Case and Levi [13], from SNMP Research, were the first to use SNMP as a delegation protocol. The task was written into a MIB of a receiving agent by a manager using the push model. The results were stored in a table of the agent, and returned to the manager. This work has inspired projects by Koojman [14] and by Williamson and Farrell [10]. This paper has used some of the ideas of the push model delegation schemes proposed by Case & Levi. Another implementation by Siegl and Transmuth's [15] uses a stack language to describe delegated tasks thus improving efficiency. The prototype from TU Braunschweig [16] used the pull method to transfer management procedures.

To develop more powerful management approaches, the Distributed Management group (DISMAN) was charted by IETF in 1996 to come up with standards. Their work has followed three different approaches (i) further development of the Manager-to-Manager (M2M) MIB (draft stage) (ii) Script MIB (continuation of Case and Levi's protoype and Goldszmidt's MbD ideas) and Scheduling MIB (both released in Oct 99) and (iii) MIB's for specific management functions that can be invoked on remote devices (draft stage). Jasmin [17] is a Java implementation of the Script MIB developed by the MIB authors. Other implementations using mobile code are in [18].

AgentX [19] provides a method to distribute MIB variables among subagents, thus distributing agent's tasks. The focus of this paper is the distribution of manager's tasks. AgentX can be seen as a complement rather than as an opposite approach to the distribution of monitoring functions.

6 Conclusions & Future Work

This paper describes a distributed scheme for a monitoring system for large heterogeneous distributed computing systems, with the purpose of overcoming the scalability problem. The solution proposed and implemented uses the SNMP network monitoring protocol, thereby ensuring ease of implementation, interoperability and acceptability, and is based on the most recent RFC standards proposals (RFC 2592). The entity which supports distribution, called ILM, is easily integrated with the remaining modules of the SIMONE monitoring system.

The performance of the monitoring system is evaluated by observing the delay involved in performing a designated monitoring task, in this case obtaining the least loaded machine in the target system. Experimental results indicate the rapidly deteriorating performance (large delays) of the centralized system for target sized beyond 100-200. The distributed scheme alleviates the problem and the actual performance varies with the extent of distribution (strongly distributed performs better than weakly distributed). Within this scale of distribution, an optimal configuration is observed for a balanced configuration, where the number of ILM's managed by the top level manager is the same as the number of agents managed by a ILM.

The distributed version of SIMONE can be configured for multiple layers of ILM's. More comprehensive experiments are needed to evaluate the performance of different configurations with multiple levels. A future enhancement under development envisions the ability to dynamically choose and operate in a desired configuration, that should be optimal for the operating conditions. This re-configurable system would also provide some degree of fault-tolerance by reassigning managerial functions in the case of failure.

Acknowledgements

This work was partially funded by the National Science Foundation under grant EIA-9975275 and by the Comision Interministerial de Ciencia y Tecnologia (Spain) under grant TIC98-1162-C02-02. Dr. Miguel-Alonso's stay at Purdue University is also supported by the Secretaria de Estado de Universidades, Investigacion y Desarrollo (Spain).

References

[1] M.T. Rose, *The Simple Book: An Introduction to Management of TCP/IP - based internets*, Prentice Hall, 1996.

[2] Kraig Meyer, Mike Erlinger, Joe Betser, Carl Sunshine, German Goldszmidt and Yechiam Yemini, "Decentralizing Control and Intelligence in Network Management", *in Proceedings of the 4th International Symposium on Integrated Network Management*, May 1995.

[3] Rajesh Subramanyan, José Miguel-Alonso and José A.B Fortes, "Design and Evaluation of a SNMP-based Monitoring System for Heterogeneous, Distributed Computing", Technical Report, TR-ECE 00-11, School of Electrical and Computer Eng., Purdue University, July 2000.

[4] R. Wolski, "Dynamically Forecasting Network Performance using the Network Weather Service", *Cluster Computing*, vol. 1, pp. 119–131, 1998.

[5] B. Tierney, W. Johnston and B. Crowley, "The Netlogger Methodology for High Performance Distributed Systems Performance Analysis", *in Proceedings of 7th IEEE International Symposium on High Performance Distributed Computing*, pp. 260–267, 1998.

[6] Craig A. Lee, James Stepanek, Rich Wolski, Carl Kesselman and Ian Foster, "A Network Performance Tool for Grid Environments", *in Proceedings of 7th IEEE International Symposium on High Performance Distributed Computing*, pp. 260–267, 1998.

[7] D.B Levi and J. Schonwalder, "Script MIB. Definitions of Managed Objects for the Delegation of Management Scripts", RFC 2592, Oct 1999.

[8] Nancy Miller and Peter Steenkiste, "Collecting Network Status Information for Network-Aware Applications", *in Proceeding of Infocom 2000*, 2000.

[9] Grid Performance Working Group, Grid Forum, "White Paper: A Grid Monitoring Service Architecture (Draft), http://www-didc.lbl.gov/GridPerf/papers/GMA.pdf".

[10] Bradley Williamson and Craig Farrell, "Distributed Management using the Remote Monitoring Management Information Base (RMON MIB) and extensible agents", Technical Report, Curtin University of Technology, September 1996.

[11] J-P Martin-Flatin, S. Znaty and J-P Hubaux, "A Survey of Distributed Network and Systems Management Paradigms", EPFL, SSC, Lausanne, Switzerland http://sscwww.epfl.ch Aug 1998.

[12] Y. Yemini, G. Goldszmidt and S. Yemini, "Network Management by Delegation", *in Proceedings of the International Symposium on Integrated Network Management*, pp. 95–107, May 1991.

[13] J.D. Case and D.B. Levi, "SNMP Mid-Level-Manager MIB", *in Internal Draft 00, SNMP Research*, October 1993.

[14] R. Koojman, "Divide and Conquer in Network Management using Event-driven Network Area Agents", Technical Report, TU Delft, Mar 1995.

[15] M.R. Siegl and G. Transmuth, "Hierarchical Network Management- A Concept and its Protoype in SNMPv2", *in Proceedings 6th Joint European Networking Conference*, May 1995.

[16] J. Schonwalder, "Using Multicast-SNMP to Coordinate Distributed Management Agents", *in Proc. IEEE Workshop on System Management*, July 1996.

[17] Aiko Pras and Jurgen Schonwalder (Editors), "The Simple Times", The Quarterly Newletter of SNMP Technology, Comments and Events, Nov 1999.

[18] Nikolas Anerouisis, "An Architecture for Building Scalable, Web-based Management Services", *in Special Issue on Enterprise Management*, Mar 1999.

[19] M. Daniele, B. Wijnen and D. Francisco, "Agent Extensibility (AgentX) Protocol", RFC 2257, January 1998.

[20] J. Schonwalder, "Network Management by Delegation- From Research Protoypes Towards Standards", *in Computer Networks and ISDN Systems*, pp. 29(15):1843–1852, November 1997.

[21] G. Goldszmidt, "Distributed Management by Delegation", *Ph.D Thesis, Columbia University*, vol. , Dec 1995.

[22] M. Sloman, "Management Issues in Distributed Services", *in Proceedings IEEE Workshop on Services in Distributed and Networked Environments, (SDNE '95)*, 1995.

[23] R. Jain, *The Art of Computer Systems Performance Analysis*, John Wiley & Sons, 1991.

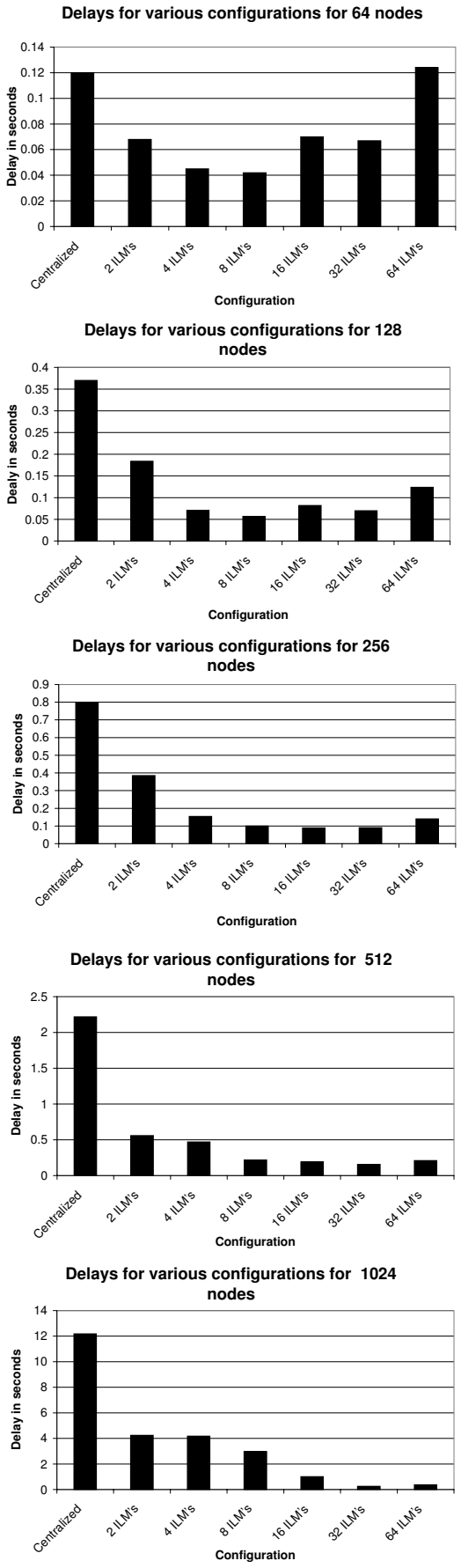

Figure 8: Delays for different distributed configurations