7-1-2000

Design and evaluation of a SNMP-based monitoring system for heterogeneous, distributed computing

Rajesh Subramanyan
Purdue University School of Electrical and Computer Engineering

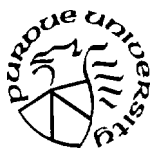
José Miguel- Alonso
Purdue University School of Electrical and Computer Engineering

José A. B. Fortes
Purdue University School of Electrical and Computer Engineering

DESIGN AND EVALUATION OF A SNMP-BASED MONITORING SYSTEM FOR HETEROGENEOUS, DISTRIBUTED COMPUTING

RAJESH SUBRAMANYAN
JOSÉ MIGUEL-ALONSO
JOSÉ A. B. FORTES

TR-ECE 00-11
JULY 2000



SCHOOL OF ELECTRICAL
AND COMPUTER ENGINEERING
PURDUE UNIVERSITY
WEST LAFAYETTE, INDIANA 47907-1285

Design and evaluation of a SNMP-based monitoring system for heterogeneous, distributed computing

Rajesh Subramanyan, JosC Miguel-Alonso and JosC A.B Fortes
School of Electrical and Computer Engineering
1285 Electrical Engineering Building
Purdue University
West Lafayette, IN 47907-1285
{subraman,miguel,fortes}@ecn.purdue.edu

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).

Contents

1 Introduction **1**

2 Characterization of SIMONE **3**

3 Implementation Objectives and Monitoring Techniques **5**

4 The Architecture of SIMONE **6**

5 Performance Indicators and their Measurement **9**

6 Experiments, Results and Interpretation **12**
6.1 Best Effort Resolution 13
6.2 Realistic Resolution 15
6.3 Latency 15
6.4 CPU Intrusion. 16
6.5 Communication Overheads 17

7 Related Work **17**

8 Conclusions and current work **18**

List of Tables

1 Tlne list of performance parameters obtained by SIMONE 3

2 Characterizing SIMONE with reference to other monitoring systems. The second column shows possible characteristics according to the criteria listed in the first column. The third column shows the attributes that best characterize SIMONE. 5

3 Workstations in the testbed. 13

4 Best effort resolution for manager request (MR), manager update (MU), and agent (A) scenarios. All resolution values are in ms. 14

5 Best effort resolution for manager-agent (MAP) scenario for local and remote agent. All resolution values are in ms. 14

6 Realistic resolution on a local agent. Resolution values are computed for different AUP's. RR is the best effort resolution (BER) value when BER is equal to AUP. The manager and agent reside in the machine aulll. Units are in ms unless specified otherwise. 14

7 Latency on local and remote agent in ms. Local agent is hermes and remote manager-agent pair is aulll-hermes. Latency is measured as the round trip time. 16

List of Figures

1 Functional layers of SIMONE 6
2 Functional modules that implement the functional layers of SIMONE. Each module is a collection of one or more programs. 7
3 Mechanism of a request and response (a) in SNMP-based monitoring and (b) using UNIX commands 10
4 Illustration of resolution: (a) generalized resolution (b) manager request scenario (c) manager update scenario (d) agent scenario (e) resolution of a manager-agent pair 11
5 Method of computing realistic resolution. 14
6 Realistic resolution for (a) local agent hermes and (b) remote agent itul11-hermes 15
7 (a) CPU intrusion for BER-MAP (local) for host aulll and (b) Communication overheads 16

Abstract

This paper describes and evaluates a tool to measure, compute and display performitnce data of any machine in a wide-area, heterogeneous, distributed computing system. The monitoring tool is easily deployable, easily extensible, and supports various degrees of centralization without significant redesign effort. The tool leverages widely used network protocols (SNMP) for communication, thus being applicable to many distributed systems.

The paper discusses the design and development of a monitoring system (SIMONE) and the performance studies conducted to evaluate the tool. SIMONE; consists of a manager which requests, receives, and processes data from individual machines or hosts and presents the results to the user. The manager is designed to measure a set of performance parameters determined useful in a network-computing environment. The hosts of the target system run daemons which service requests from the manager. The reply to each request consists of the variable values obtained from the host. Performance measurements carried out on the prototype SIMONE are reported and compared to similar measurements using alternate monitoring methods. Performance metrics include resolution of monitored measurements, latency between data request and presentation, and communication and CPU overheads. The performance of SIMONE shows significant improvement (better resolution, less latency, lower overhead) over that of alternate monitoring methods.

1 Introduction

Advances in network technology have enabled integration of geographically distributed, heterogeneous resources (hereinafter called hosts) into systems capable of providing computational cycles on demand. Examples include systems referred by different narnes such as metacomputing, network-computing, ubiquitous computing [9], and grid-based computing [11], i.e., systems like Globus [8], Legion [1], PUNCH [17] etc. For effective usage of these computing systems, it is necessary to monitor the performance of individual host components such as processor, network, disk, and memory.

Distributed computing systems pose several additional monitoring problems over centralized systems, making the design of a monitoring system a challenging task. Potential solutions must include local monitoring subsystems that can be integrated and scale with the system, generate monitoring information at multiple levels of granularity, do not process unnecessary data, and keep overheads at acceptable levels. The description of the design of such a local subsystem is the goal of this paper.

Monitoring tools for distributed systems fall under three broad categories. Numerous tools developed in the late 80's were complete systems, but system-specific. The second class of tools rely on network management protocols (e.g. SNMP, RMON, CMIP [25]) and are primarily intended for network monitoring, but can also support limited host monitoring. Under a third category of monitoring tools are recent efforts that use information generated from UNIX system commands and socket-based TCP/IP communication.

This paper discusses the architecture, implementation, and performance of SIMONE (Simple Network Management Protocol-based Monitoring System for Network Computing), a modular monitoring system for a heterogeneous, wide-area distributed computing system. SIMONE falls under the second of the three categories described above, with emphasis on host monitoring. The design of SIMONE reflects the following goals (also refer to Section **3):**

- a Easy deployability without special access privileges
- a Easy extensibility of functionality
- a Easy adaptability to different modes of operation, e.g. centralized. vs. distributed management

It performs in an environment in which machines may be separated by large network distances, belong to different administrative domains, or operate under different operating systems. SIMONE utilizes existing and widely used management protocols (SNMP [15]) and libraries. Any machine on the Internet is a suitable candidate for monitoring.

SIMONE is particularly useful in a distributed computing environment for the following reasons:

- Information from SIMONE can be used for resource management, scheduling network-computing applications, trouble-shooting and providing performance information to network-computing applications. Examples of useful parameters measured by SIMONE are under the broad areas of general monitoring information (e.g. host description), processes running on specified hosts (e.g. CPU time, memory used by a process), machine monitoring (CPU load, average page fault rate), host interface and link information (traffic in/out of host) and monitoring overhead measurements. See Table 1 for a complete list of implemented parameters. The developed modules provide mechanisms to supply information to end users.

- Intesoperabilty and reduced implementation costs by virtue of using SNMP. SIMONE relies on locally collected information, is easily deployed, and its modules have been constructed so that large systems with different configurations can be easily constructed.

Some of the terms used in this paper are defined here. The word data, unless otherwise stated, is used to refer to data generated by the monitoring system. A host with a SNMP daemon plays the role of an agent. A variable is an entity measured by the host's agent. Several variables that exhibit some similarity are aggregated into groups and several groups form the *agent* table. Each update by the agent causes the variable value to be overwritten. A parameter, in contrast, is an entity derived from one or more variables. In SIMONE, parameter values are computed by the manager. A collection of values of a parameter for a particular machine and for a time interval is called results. The user is a human or an application which needs monitoring data and obtains it using SIMONE.

SIMONE consists of a set of modules and services that forms a two-entity system, the agent and the manager. The agent collects and forwards data upon being queried. Other functions, namely, managing, processing runtime information, and controlling the entire mon-

itoring system, are performed by the manager. Using the data retrieved from the agent(s), the manager computes a user-specified set of parameters shown in Table 1 below

General	*Process Monitoring*	*[Machine Monitoring*	*Network Monitoring*	*Overheads Monitoring*
Machine name	Max., min. & avg. CPU time	CPU load	Avg. & max. traffic in/out	Total SNMP traffic-in
Machine description	Total CPU seconds used by a process	Free cycles	Avg. & max. traffic between two machines	Total SNMP traffic-out
Machine uptirne	Total elapsed time	Avg. page-fault rate	Delay and bandwidth between hosts (proposed)	CPU intrusion
Machine location	Total memory used	Memory (total & currently available)		Memory intrusion
Machine services	Avg. & total amount of I/O generated by the process	Swap space		
Reserved for tests		Number of processes		
Reserved for tests		Computation rate		

Table 1: The list of performance parameters obtained by SIMONE

The remainder of the paper is organized as follows. Section 2 characterizes SIMONE by comparing it to other monitoring systems. Section 3 discusses the implementation objectives and the monitoring techniques used. Section 4 describes the design, architecture, and components of SIMONE, including user interfaces. Sections 5 and 6 focus on the performance of SIMONE. Experimental evaluation of SIMONE is reported and the results are compared against other monitoring methods that employ UNIX system commands. Section 7 discusses other related work. Conclusions and work in progress are presented in Section 8.

Characterization of SIMONE

Table 2 uses a list of criteria in order to compare and contrast SIMONE with other monitoring systems. The first four criteria describe a monitoring system in general while the latter four criteria compare properties of specific monitor components. SIMONE focuses on performance monitoring, in particular system monitoring [13]. Performance monitoring may be divided into the following:

1. Application monitoring, program steering, program visualization on shared-memory and distributed-memory parallel systems. Examples are Falcon [7], Pablo [5], Paradyn [2] and SIMPLE [18].

2. Network monitoring and management. There are numerous research efforts and commercial products, the latter due to their commercial application (e.g. of vendors: Tivoli,

BMC, Candle [25] etc). Research efforts have lead to well established protocols and paradigms like RMON, SNMP, CMIP [25].

3. System monitoring in distributed systems. A spate of efforts in late 80's and early 90's led to development of tools like ARTS, Kato, Jade, Incas, Tmp, ZM4, PATOP, DETOP, TOPSYS, Maritxu [10] and Jewel [6]. The tools were customized to the target system, which were distributed, cluster, or high performance computers, but in almost all cases homogeneous. Recent projects underway like Netlogger [3], NWS [21], Remos [16] and Gloperf [4] have been built for metacomputing.

4. Integrated management systems. These are large commercial products that provide a number of management services like configuration, fault, resource, accounting and performance management. Examples are enterprise systems such as SunNet Manager, HP-Openview, IBM-Tivoli [25].

Comparisons may be made between our work and the above based on the following (A) target system, (B) monitoring objective and (C) implementation techniques. With respect to the target system, SIMONE is similar to wide area systems of (2) and the recent works in (3); with monitoring objective, to **(3)** and (4) (integrated systems include system monitoring); and with respect to implementation methods to (2) and (4). Notable differences with each of the four categories are:

- With respect to to application monitoring, the target system, monitoring goals and methods all differ. However, some of the design issues and challenges are common.

- With respect to older systems for system monitoring (3: old), the target systems are different (local-area homogeneous as opposed to wide-area heterogen~eous).

Among the individual components of a monitoring system, instrumentation data are generated by a probe or a sensor, which may be implemented using software code, a hardware chip, or a combination of both (hybrid). SIMONE uses a software probe,, and is therefore classified as a software monitor [20]. Event-triggering and timer-driven (sampling) [13] are the two distinct approaches for collecting instrumented data from the target system; the selection is determined by the nature of the application of the monitoring system. SIMONE uses

Characterization Criteria	*Broad foczi of extant work*	*Specific focii of SIMONE*
Monitoring/management functionality and purpose	Accounting, fault, security, configuration performance	Performance, fault
Class of performance parameters being measured	Application, network, system	System
Target system	Stand-alone, parallel, supercomputing	Heterogeneous distributed
Intended use of monitoring	Resource management, visualization, trouble-shooting, tuning, debugging network-aware applications	Used for resource management trouble shooting scheduling network-computing applications
Data analysis and display	Real-time, delayed	Both
Data collection and processing (architecture)	Centralized, distributed, weakly distributed [10]	Centralized (not rigid)
Monitor sensor type or data extraction	Software, hardware, hybrid	Software
Instrumentation strategy	Timer/sampling, event driven/tracing	Timer

Table **2:** Characterizing SIMONE with reference to other monitoring systems. The second column shows possible characteristics according to the criteria listed in the first column. The third column shows the attributes that best characterize SIMONE.

sampling, with the user specifying the polling rate. Monitoring data could either be analyzed immediately following generation, or delayed until the collection is completed. SIMONE has both features, with the user making the selection. Although the collection and evaluation processes can be centralized or distributed, scalability demands that both processes be distributed. Distribution, however, brings up other monitoring issues and challenges [12]. The current iniplementation of SIMONE supports centralized mode although it can be easily reconfigured to a distributed mode [22].

3 Implementation Objectives and Monitoring Techniques

This section described the goals of SIMONE and how they are met by its design.

Operation in a heterogeneous environment: SIMONE obtains monitoring information using universally understood SNMP protocol to query the SNMP daemon (snmpd) which is available on most machines. Additional information not provided by snmpd is obtained through a supplement daemon (m_daemon) designed for SIMONE, which operates with normal user privileges. The implementation of m-daemon is similar to snmpd, while its running conditions are different. The m_daemon would complement the collection of daemons required to operate a network-computing system, and would be started along with them.

Information filtering and analysis: Only those variable values needed to compute user-requested parameters are retrieved and stored in the manager host. Computations are delayed until a user request for a given result is received.

Overheads: Data transfer is kept low by requesting only essential variables and performing bulk transfers whenever several variables from one group in the table is requested.

Dynamic inclusion of machines in the set of monitored hosts: Agents that have the m_daemon program installed can join the pool of monitored hosts at any time. The m_daemon process can be started from a central location, i.e., the manager, as well as directly on the individual host.

Monitor operation: The operational behavior of SIMONE is partly controlled by a set of configuration values provided by the user and an internal set pre-configured in the current implementation.

Run time visualization: The user chooses to view results computed from either old data or current data on a continuous basis, even while they are being generated. In the latter case all display mechanisms, including graphs, incorporate the latest parameter values computed in the most recent update.

The Architecture of SIMONE

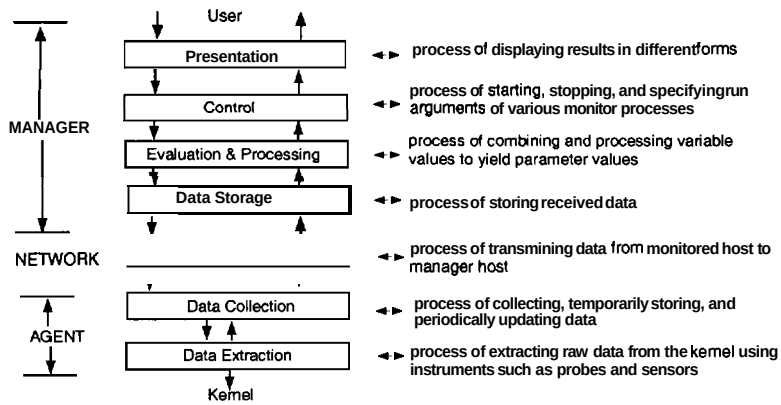


Figure 1: Functional layers of SIMONE

Figure 1 illustrates the layered architecture of SIMONE. Figure 2 is a detailed view of the modules that constitute each layer, starting with the agent.

Agent Components

- Data extraction layer: The two daemons (snmpd and m-daemon) of an active agent retrieve monitoring data from the kernel. Some are defined in RFC 1213 and RFC 1514, others are non-standard and hence user defined. The configuration determines the capture or update period, i.e. the time period of overwriting the variable values with new monitoring data. The daemons service requests by generating SNMP packets with current values of the variable or variables (for bulk requests). Snmpd cannot be controlled by non-root users. In contrast, the user has full control of m-daemon, starting and stopping remotely, adding to the list of variables gathered, and controlling its running by varying agent configuration variables such as update period. Consequently, the experiments (see Sections 5 and 6) are conducted with m-daemon variables.

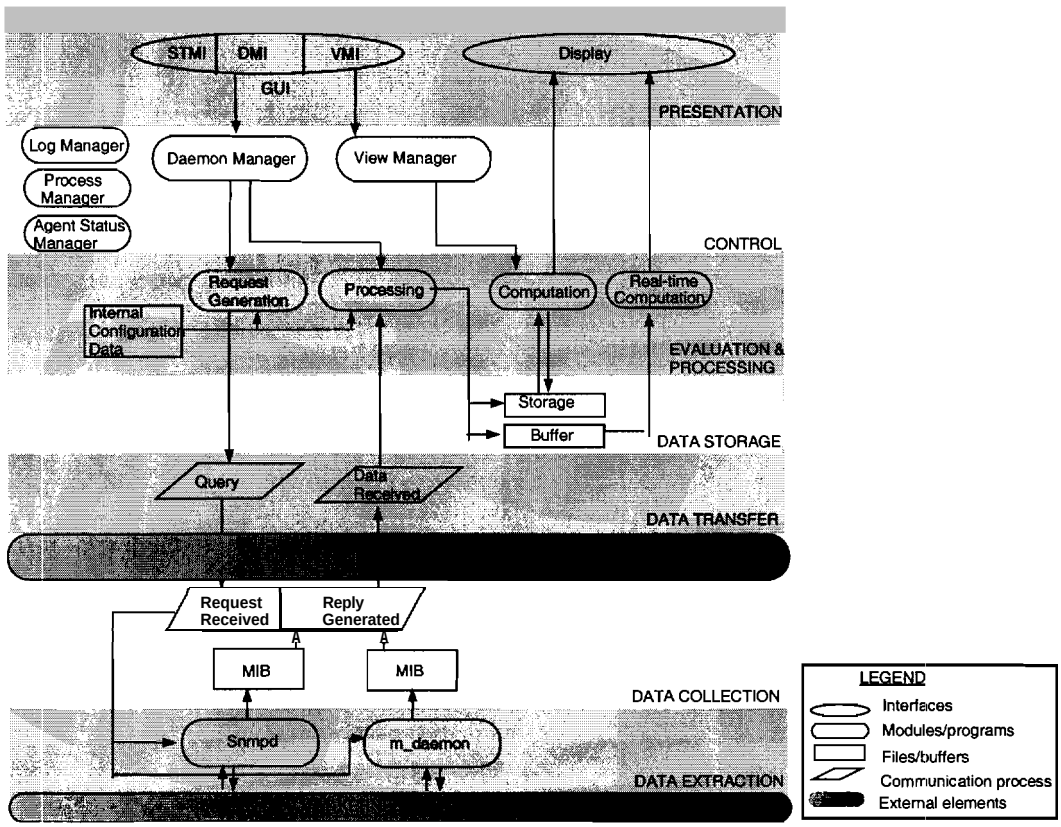


Figure 2: Functional modules that implement the functional layers of SIMONE. Each module is a collection of one or more programs.

- Data collection layer: The variable values are stored in a virtual table whose data structure is specified by the MIB of the corresponding daemon. The agent host does not perform any

computation as is the practice in SNMP systems [15].

Network Component

a Transfer layer: This layer encompasses the transfer of data through variable requests sent by the manager and replies sent by agents. The time interval between manager queries (polling rate) is set by the user.

Manager Components

a Storage layer: In the current implementation, the storage of data is centralized. The data is stored in buffers or log files (depending on short term or long term) in the manager. The SIMONE user determines both the data to be stored and the parameter/parameters to be computed from this data, thus reducing CPU and storage overheads.

a Evaluation and processing layer: The core functions of SIMONE are divided into (1) instrumentation for collection and (2) analysis and evaluation, both controlled by user requests. The former is the process of periodically collecting data from the agents., parsing and extracting relevant information and storing it. In the latter, a module (Figure 2) processes the required data from storage and computes appropriate parameter values followed by the generation of displays.

a Control layer: The function of the Daemon Manager (DM) is to generate requests to be sent to the agent in order to collect data. The View Manager (VM) controls the generation of results computed from the data logs. The Process Manager (PM) helps manage the monitoring processes and the Log Manager (LM) similarly manages log files and also allows the user to delete log files. The Agent Status Manager (SM) helps determine which agents are active and starts agent daemons (only m_daemons).

a Presentation layer: The user interface component of SIMONE helps in the control and operation of the monitor and provides the flexibility of setting input operation values. The two most important interfaces are the Daemon Manager Interface (DMI) which allows the user to specify the machines and parameters to be monitored, and the View Manager Interface (VMI) that allows the user to request which results should be computed. The output is displayed in the form of result summaries, tables, graphs etc. Interfaces are also provided for

LM, PM, and SM units of the control layer. Presently, a fairly extensive user-friendly GUI is provided. Work is in progress on a library-based API.

5 Performance Indicators and their Measurement

The performance study of SIMONE has two broad focii: (1) performance of specific entities or subsets of the monitoring system and (2) performance of the complete monitoring system. The two distinct entities of the monitoring system, the agent and the manager, are separately evaluated. Regarding performance of the complete system, test:; are limited to a single monitor-agent pair. Scalability issues are an important concern in the performance of the complete monitoring system, and are currently being studied (Section 8). However, the indicators defined and measured in this paper have a bearing on the performance of a complete, large-scale system. This makes the performance of individual modules, components, and subsystems a necessary condition for obtaining a scalable system.

The performance indicators which are used to characterize the monitoring system are (1) resolution (2) latency, and (3) communication and CPU overheads. The term *observation* is used to denote the act of gathering the MIB variable values from an agent by the manager. In contrast, the term *measurement* denotes monitor performance as observed in the evaluation experiments. Measurements are made on SIMONE (referred hereinafter as SNMP-based monitoring) and compared with those made using traditional UNIX monitoring commands (specifically, top and `rup`).

For the purpose of evaluation and comparison of alternate approaches, it is convenient to think of SIMONE in terms of the two main entities: the manager and the agent(s). These two software components communicate via MIB variables which are asynchronously read and written by the manager and agent, respectively. This is depicted in Figure 3(a). The two software components could be on the same or on different machines. The agent is accordingly referred to as a local or remote agent. In the latter case, SNMP communication is via the network.

Although the command top is primarily intended to run on a local environment, remote login (rlogin) or remote execution (rexec) can be used to allow a front-end machine to retrieve the performance measurements of other machines, with communication performed via TCP.

Rup is specifically designed to monitor a remote machine. A user can involse this command from one machine (the a front-end, playing the role of the manager) to monitor a remote one (the **rup** server, playing the role of the agent). Communication between a front-end and **rup** server is performed using RPC (Figure 3(b)).

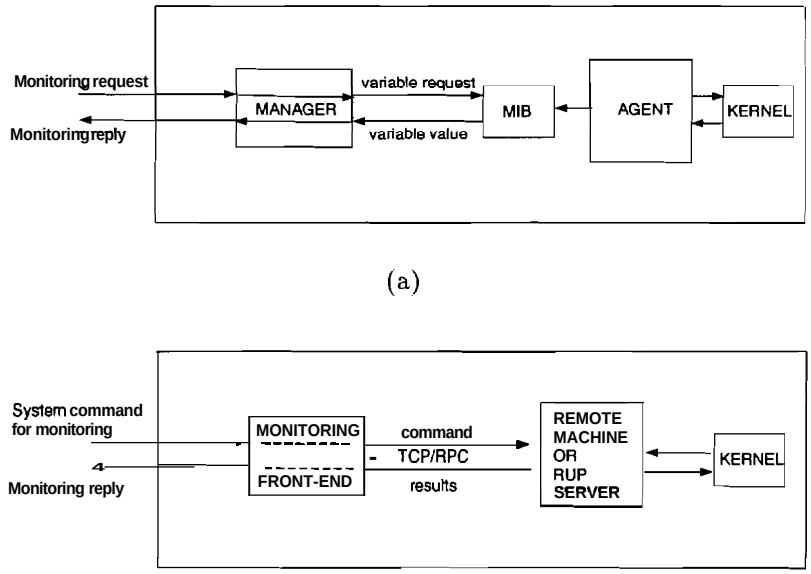


Figure 3: Mechanism of a request and response (a) in SNMP-based monitoring and (b) using UNIX commands

The two input settings of SIMONE that have a fundamental impact on SIMONE behavior and, thereby, on the delivered performance are (1) the **Agent Update Period (AUP)** or scheduled time intervals at which variable values in the agent's MIB are updated and (2) the **Manager Polling Period (MPP),** or scheduled time intervals between the generation of two manager requests.

Measurements are performed over three basic scenarios (agent, manager request, and manager update) and over a manager-agent pair (see Figure 4 for details). For each scenario, an *input period* (the time interval between two actions requested to the module) is set, and an *output resolution* (the time interval between two responses generated by the module) is measured. We define **Best Effort Resolution (BER)** as the shortest time interval between two responses, when requests are received at minimum time intervals. The BER gives the maximum rate (speed) at which a module responds for short intervals of time (i.e peak rate).

The following indicators are, thus, measured:

1) **BER-A** for the agent. Inputs are manager requests, and outputs are responses with MIB variable values.

2) **BER-MR** for the manager request module. The input is the MPP, while outputs are the generated SNMP agent requests.

3) **BER-MU** for the manager update module. Inputs are the responses received from the agents; outputs are parameter updates.

4) **BER-MAP** for a manager-agent pair. The input is the MPP, for a given AUP; outputs are parameter updates.

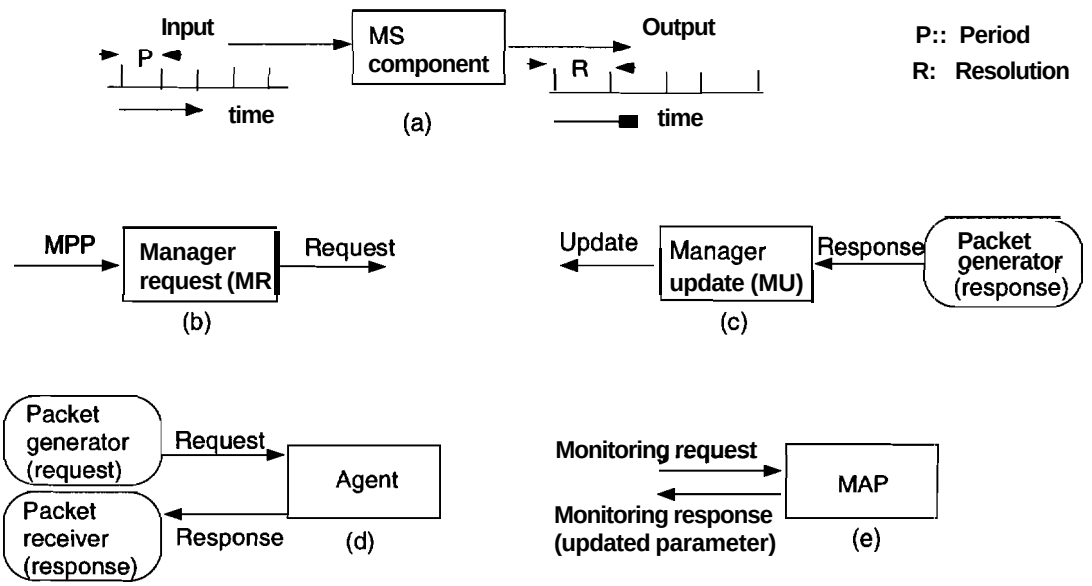


Figure 4: Illustration of resolution: (a) generalized resolution (b) manager request scenario (c) manager update scenario (d) agent scenario (e) resolution of a manager-agent pair

Another indicator defined in the evaluation of a manager-agent pair is the **Realistic Resolution (RR-MAP).** The manager updates the monitoring parameter!; when a response from the agent is received. Receiving two consecutive responses does not imply that these responses are different: the second one could yield exactly the same information as the first one as a consequence of the AUP being too long. RR can thus be defined as the shortest time difference between two consecutive parameter updates which are processed from agent responses with updated variables. RR captures the accuracy of observations.

Besides resolution, the following indicators are measured:

- **Latency:** the elapsed time between sending a request and receiving a response.
- **Overheads or intrusion:** the degradation of performance of the computing system due to the sharing of resources (CPU, communication channels) with the monitoring system. In this paper, incremental CPU loads and monitoring packets are measured.

The accuracy of the values of variables measured by SIMONE depends on the accuracy of snmpd and Unix commands, and on the delay (round trip time + agent service time + manager service time + AUP) caused by SIMONE while obtaining the information. Effort has been made in SIMONE in keeping the delays low, and experiments are focused on this aspect under the worst case scenarios. In particular, the Realistic Resolution provides insight into the accuracy of the measurement. The accuracy required by the user is application-dependent.

6 Experiments, Results and Interpretation

The experimental testbed for determining performance indicators consists of six Sun workstations (refer to Table **3)** connected by a local area network and running the Solaris 5.6 operating system. Four of the machines are located nearby and belong to one subnet, and the other two belong to another one. For a fair comparison between the two monitoring approaches, the tests were conducted under similar load conditions, and measurements for the two approaches were alternated. Both monitoring systems were assigned the job of observing the CPU time consumed by any one user-level process that is not part of the processes being evaluated. Simple devices such as dummy agent, packet receiver and packet generator are implemented to aid measurement process. Conducting the experiments on a LAN helps better evaluate the objective of worst case measurements (stress tests). The worst case scenario is exhibited when network distances are low, and the intervals between packets are small.

In the performance tests, the agent AUP values range between 1 and 500 ms. They are shorter than default values, which range from seconds to a few minutes. Small AUP values increase the daemon load, which in turn affects the agent service time. The SNMP test values are hence more conservative than what might have been obtained running the agent at

Machine	Configuration	Processor Clock Speed.	Subnet
aulll	Ultra-5	300 Mhz	subnet1
hermes	Ultra-l	167 Mhz	subnet1
alx27	SPARCstation-5	70 Mhz	subnet1
athena	SPARC-LX	50 Mhz	subnet1
drum	SPARCstation-5	70 Mhz	subnet2
whitford	SPARCstation-5	70 Mhz	subnet2

Table 3: Workstations in the testbed.

standard configuration values. Since our goal was performance-testing, worst-case scenario values are desirable. However, at short AUP's (below 100 ms), the agent is an artificial bottleneck in MAP scenarios, completely concealing the performance of the manager. Thus, an intermediate value of AUP of 500 ms is considered reasonable for all performance tests. As AUP can be modified only in m_daemon, the experiments were performed using this daemon.

6.1 Best Effort Resolution

A large number of inputs (1500) is generated at minimum intervals of time and the time between sending the first input and receiving the last output is recorded. BER values are affected by processor speed and AUP values, the latter when the agent is a part of the scenario. Measurements were made for different AUP's and processors for different scenarios, and measurements recorded in Tables 4 and 5. BER gives the peak rate at which output can be generated by a component (agent or manager) for short intervals of time. The performance of both manager and agent is ascertained irrespective of which one of the two is most likely to be the bottleneck. Load increases due to shorter AUP's are more pronounced in low-speed processors. BER-MAP values for SNMP are largely affected by low values of AUP, processor speed, as well as network distance if the agent is remote. To demonstrate the effect of network distances, two machines (drum and alx27) with the same processor speed, but located on different subnets, are chosen. The manager machine aulll is on the same subnet as alx27, while aulll and drum are on different subnets. Measurements indicate that BER values for SNMP are 2-20 times better than those for UNIX commands.

hermes	
drum	8.27

Manager	*BER-MU*
au111	3.85
hermes	5.38
drum	20.77

	BER-A			
	AUP			
Agent	*1 ms*	*10 ms*	*100 ms*	*500 ms*
aulll	135.0	76.43	14.29	8.57
hermes	91.43	80.71	14.29	5.71
drum	105.0	77.14	15.0	10.0

Table 4: Best effort resolution for manager request (MR), manager update (MU), and agent (A) scenarios. All resolution values are in ms.

	BER-MAP (local)					
	SNMP AUP				**UNIX**	
Manager-Agent	*1 ms*	*10 ms*	*100 ms*	*500 ms*	Top	Rup
aulll	61.33	35.33	6.67	4.0	252.01	70.7
hermes	58.67	37.3	14.67	5.71	326.7	77.3
drum	189.33	191.33	108.67	57.3	934.0	208.0

	BER-MAP (remote)			
	SNMP AUP		**UNIX**	
Manager-Agent	*1 ms*	*500 ms*	Top	Rup
au111-hermes	157.3	9.3	1815.33	98.67
au111-alx27	288.67	25.33	3440	93.33
au111-drum	504.67	38.67	2860	198.67
aulll-athena	494	58.67	5580.0	88.67
drum-whitford	906.0	56.67	1664.0	348.67
alx27-au111	164.0	28.0	1733.3	206.67
drum-au111	210.67	62.67	1680.0	633.74
athena-au111	272	20.67	1606.67	400.0

Table 5: Best effort resolution for manager-agent (MAP) scenario for local and remote agent. All resolution values are in ms.

	AUP								
MPP	*1 ms*	*5 ms*	*10 ms*	*20 ms*	*50 ms*	*100 ms*	*1 s*	*10 s*	*60 s*
1 ms	67.3	36	26	22	13.3	9.3	4	3.6	3.3
Max(AUP,Res)	67.3	36	26	**Min=22**	50	100	1 s	10 s	60 s
5 ms	88	78.7	78.7	70	36	24	13.3	10.7	10.3
Max(AUP,Res)	88	78.7	78.7	70	**Min=50**	100	1 s	10 s	60 s
Min(1 ms, 5 ms)				**22 ms**					

Table 6: Realistic resolution on a local agent. Resolution values are computed for different AUP's. RR is the best effort resolution (BER) value when BER is equal to AUP. The manager and agent reside in the machine aulll. Units are in ms unless specified otherwise.

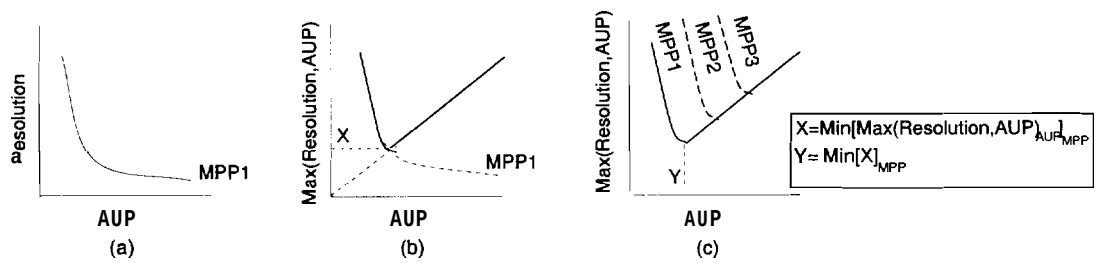


Figure 5: Method of computing realistic resolution.

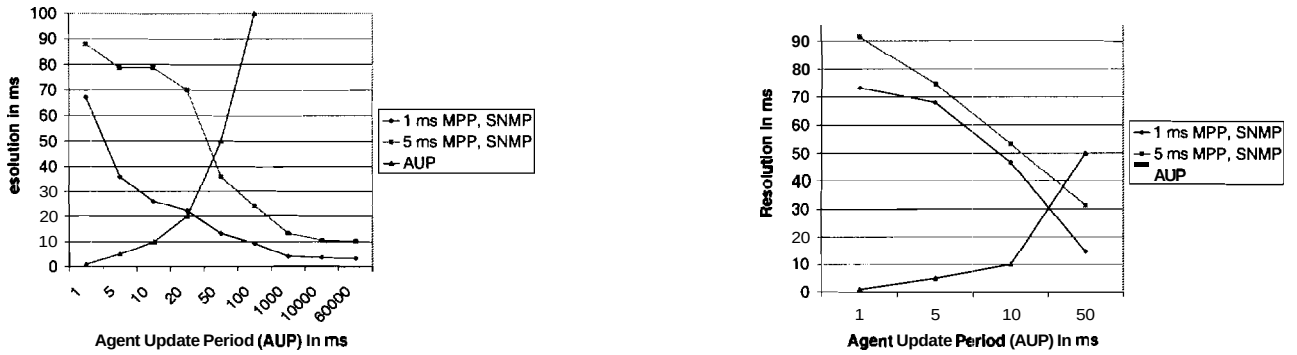


Figure 6: Realistic resolution for (a) local agent hermes and (b)remote agent aulll-hermes

6.2 Realistic Resolution

Realistic resolution has been defined for a MAP scenario alone and is obtained by measuring resolution for different sets of MPP and AUP. Figure 5 describes the sequence of computing RR from resolution values. Previous experiments indicate that resolution decreases with AUP for fixed values of MPP. This is shown in Figure 5(a). Figure 5(b) shows Max(Resolution, AUP) vs. AUP for a fixed MPP and is obtained from Figure 6(a). Figure 5(c) shows Max(Resolution, AUP) vs. AUP for different MPP's. Thus it is seen that RR= Min(Max(Resolution, AUP)) is the intersection point of the resolution curve and the AUP line, when MPP is the minimum. RR values for local and remote agents were computed as 20 ms and 30 ms, respectively, from Table 6 and Figure 6(c). RR signifies the smallest resolution at which the manager can obtain meaningful data (received responses always contain updated values).

6.3 Latency

Latency is the round trip time between the monitoring request and monitoring response. If the agent is remote, the network causes an additional delay. Table 7 shows: latency for local and remote agents. AUP, which determines service time of agent, affects the latency value significantly. Latency for SNMP at AUP=500 ms is significantly lower than that for UNIX

methods, while at minimum AUP the latency values are marginally better than for **rup.** The high latency values obtained for remote execution of top are due to the overheads of the authentication process.

	AUP				UNIX	
System	1 ms	10 ms	100 ms	500 ms	rup	top
Local	52.0	33.33	10.67	5.33	68.67	316.0
Remote	133.33	46.67	10.67	9.33	71.33	1613.33

Table 7: Latency on local and remote agent in ms. Local agent is hermes and remote manager-agent pair is aulll-hermes. Latency is measured as the round trip time.

6.4 CPU Intrusion

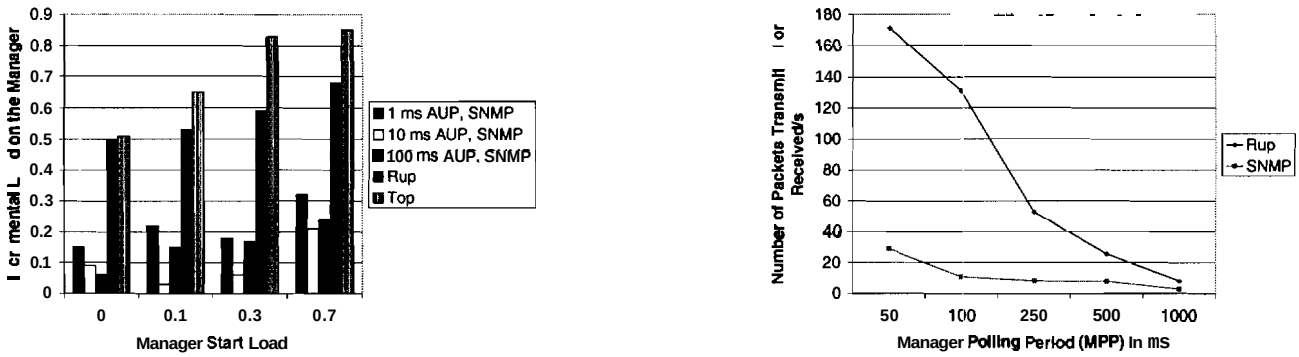


Figure 7: (a) CPU intrusion for BER-MAP (local) for host aulll and (b) Communication overheads

Increase in CPU load due to monitoring activity is measured by noting the start and end loads during BER-MAP (local) experiments on machine aulll. A synthetic load is used to obtain different start loads. It is observed that the incremental load is lower for smaller start loads. The incremental load measured for different start loads is shown in Figure 7(a). The SNMP system performs significantly better than UNIX commands with regard to CPU intrusion.

6.5 Communication Overheads

The MPP is varied and the packets received and sent by a host are observed at intervals of 5 seconds. All tests were conducted at low load when extraneous processes would not affect the tests by generating packet traffic on their own. The difference in the average number of packets received during the monitoring session and prior to it indicates the number of packets generated by the monitoring process under test. The average traffic per second computed for the SNMP and **rup** command for different polling periods are shown in Figure 7(b). SNMP system has less traffic than **rup** by a factor of 8.

Related Work

Several monitoring systems, both research and commercial, have been developed for different target systems. There are numerous commercial SNMP packages like SunNet Manager, HP-Openview, BMC, Tivoli etc., which are used for network management by different end-users (e.g. by large corporates, ISP's etc.). These packages cannot be readily reused to monitor network-computing systems because different variables need to be measured and different monitoring information needs to be generated from these variables.

Projects related to SIMONE that have been designed for similar environments but differ in objectives and implementation are NWS [21], Netlogger [3], Remos [16] and Gloperf [4]. The NWS provides dynamic resource performance forecasts in network-computing environments (Globus). Network performance, system forecasts for latency/bandwidth and average CPU time are measured using UNIX commands like **vmstat** and **uptime.** 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. Both sensing and communication techniques of SIMONE are different from that of NWS. Globus' Gloperf tool works like NWS, and periodically schedules end-to-end tests to retrieve latency and bandwidth information. A hierarchy of groups is used for measurement which makes it more complex, but the benefit is that the number of tests is reduced. Netlogger is a trace-based system used mainly for real-time diagnosis of performance problems by measuring network, host and application parameters in complex heterogeneous systems using a java-based agent to gather variable values and TCP/IP sockets for communication.

Remos 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. 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. Although SIMONE does not currently perform end-to-end tests, it is proposed in future to passively measure delays between two hosts using techniques described in [14].

While SIMONE monitor hosts on an individual basis, it is sometimes required to have a global view of a cluster (a tightly-coupled collection of machines behaving as a single compute node). Monitoring systems such as PARMON [23], SCMS [19] and SyMON [24] provide this single view of a cluster. By interfacing SIMONE with this kind of software, it would be possible to monitor a cluster as a single entity.

8 Conclusions and current work

Some aspects of the design of SIMONE are constrained by a number of important goals. The implementation and usage without requiring privileged access is of primary importance. Other implementation issues considered in the design are user flexibility, ease of expansion, and the flexibility for the user to choose and guide SIMONE operation. Resolution, latency, and overheads are the performance indicators which are used to compare the performance of SIMONE and UNIX commands.

Performance results indicate that the SNMP monitoring system performs better than comparable UNIX methods wherever comparisons were made (i.e. BER-MA, CPU intrusion, latency and communication overheads). Particularly significant is the lower CPU intrusion, an important issue in monitoring systems.

Work currently in progress considers the distribution of management functions of SIMONE among a collection of intermediate-level managers. Strategies to deploy these intermediate-level manager's are being experimentally evaluated on large testbed. The objective is to remove potential bottlenecks that arise when a centralized manager is used in a very large setup. Details of this work are provided in [22].

References

[1] A. Grimshaw, W. Wulf. Legion - A View from 50,000 feet. In *Proceedings of 5th IEEE International Symposium on High Performance Distributed Computing,* pages 89–99, 1996.

[2] A. Waheed, D.T. Rover and J.K. Hollingsworth. Modeling, Evaluation, and Testing of Paradyn Instrumentation System. Technical Report, Michigan State University, April 1996.

[3] 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,* pages 260–267, 1998.

[4] 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,* pages 260–267, 1998.

[5] D.A. Reed, R.A. Aydt, R.J Noe, P.C. Roth, K.A. Shields, B.W. Schwartz and L.F. Tavera. Scalable Performance Analysis: The Pablo Performance Analysis Environment. In *Scalable Parallel Libraries Conference, IEEE Computer Society,* 1993.

[6] F. Lange, R. Kroeger, M. Gergeleit. JEWEL: Design and Implementation of a Distributed Measurement System. Technical Report, German National Research Center for Computer Science (GMD), 1992.

[7] Gu, Eisenhauer, Kraemer, Schwan, Stasko and Vetter. Falcon: On-line Monitoring and Steering of Large-Scale Parallel Programs. Technical Report, Georgia Institute of Technology, 1994.

[8] I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputer Applications,* 11:115–128, 1997.

[9] I. Foster and S. Tuecke. Enabling Technologies for Web-Based Ubiquitous Supercomputing. Technical Report, Argonne National Laboratory, 1996.

[10] 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.

[11] I. Foster C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure.* Morgan Kaufmann, 1999.

[12] M. Mansouri-Samani and M. Sloman. Monitoring Distributed Systems (A Survey). Technical Report, Imperial College, April 1993.

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

[14] M.Stemm, S. Seshan and R. Katz. Spand: Shared Passive Network Performance Discovery. In *USENIX Symposium on Internet Technologies and Systems*, June 1997.

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

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

[17] N.H. Kapadia and J.A.B. Fortes. On the Design of a Demand-Based Network-Computing System: The Purdue University Hubs. In *7th IEEE International Symposium on High Performance Distributed Computing,* pages 89–99, July 1998.

[18] P. Dauphin, R. Hofmann, R. Klar, B. Mohr, A. Quick, M. Siegle and F. Sotz. ZM4/SIMPLE: a General Approach to Performance-Measurement and Evaluation of Distributed Systems. In *Readings in Distributed Computing Systems.* 1992.

[19] P. Uthayopas, S. Phaisithbenchapol and K. Chongbarirux. Building a Resources Monitoring System for SMILE Beowulf Cluster. In *Proceeding of High Performance Computing, Asia '99,* 1999.

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

[21] R. Wolski. Dynamically Forecasting Network Performance using the Network Weather Service. *Cluster Computing,* 1:119–131, 1998.

[22] Rajesh Subramanyan, José Miguel-Alonso and José A.B Fortes. A Scalable SNMP-based Distributed Monitoring System for Heterogeneous Network Computing. In *Proceedings of Supercomputing2000,* 2000.

[23] Rajkuma Buyya. PARMON: A Portable and Scalable Monitoring System for Clusters. *International Journal on Software: Practice & Experience, John Wiley & Sons,* 2000.

[24] Sun Microsystems. *Solstice SyMON* 1.1 *User's Guide.* Palo Alto, 1996.

[25] U. Black. *Network Management Standards.* McGraw-Hill, 1994.