1-1-2002

Toward a Reconfigurable SNMP-based Distributed Monitoring Mechanism for Networked Computing Systems

Rajesh Subramanyan

Jose Miguel-Alonso
University of the Basque Country UPV/EHU

Jose A.B. Fortes
University of Florida

TOWARDS A RECONFIGURABLE SNMP-BASED DISTRIBUTED MONITORING MECHANISM FOR NETWORK COMPUTING SYSTEMS

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

TR-ECE 02-01
MARCH 2002

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

Towards a Reconfigurable SNMP-based Distributed Monitoring Mechanism for Network Computing Systems

Rajesh Subramanyan
School of Electrical and Computer Engineering
1285 Electrical Engineering Building
Purdue University
West Lafayette, IN 47907-1285
subraman@ecn.purdue.edu

José Miguel-Alonso
Dept. of Computer Architecture and Technology
University of the Basque Country UPV/EHU
San Sebastian, Spain

and

José A.B Fortes
Dept. of Electrical and Computer Engineering
University of Florida
Gainesville, FL

This work was partially funded by the Spanish MCYT under grant TIC2001-0591-C02-02.

Contents

List of Tables

List of Figures

Abstract

The dynamic nature of large-size Network Computing Systems (NCSs) and the varying monitoring demands from the end-users pose serious challenges for monitoring systems (MSs). A statically configured MS initially adjusted to perform optimally may end performing poorly.

A reconfiguration mechanism for a distributed MS is proposed. It enables the MS to react to changes in the available resources, operating conditions, and monitoring requirements, while maintaining high performance and low monitoring overheads.

The distributed MS is organized as a tree, consisting of managed nodes running agents, one or more levels of intermediate-level managers (ILMs), and a top-level manager (TLM) for overall control. A localized decision process involves two adjacent ILM levels. The current values of a local node performance parameter called *temperature* are used in determining the transformations (merge, split, migrate) for each ILM. The implementation uses SNMP primitives for easy integration in SIMONE, a distributed SNMP-based monitoring system.

The interactions between the MS elements and different classes of jobs are studied by defining a queuing model, and by evaluating different configuration schemes using simulation. Results for the static and reconfigurable schemes indicate that reconfiguration improves performance in terms of lower processing delays at the ILMs.

Keywords: monitoring, network-computer systems, reconfiguration, SNMP

1 Introduction

A Network Computing System (NCS) is a large collection of heterogeneous resources spread across several administrative domains of a wide-area network that can be marshalled to provide distributed application services to many users. We assume an NCS to be a system with a large and varying number of nodes (nodes may join or leave the NCS pool), separated by sizeable network distances with unpredictable delays, and with continuously changing monitoring requirements. A monitoring system (MS) developed for an NCS has to factor in the complexity, size, distribution, and heterogeneity of both the environment and the applications running on it. The performance of the MS for NCS becomes an issue of particular concern, both from the standpoint of accuracy and timeliness of monitoring information, as well as the ability to impose low overheads. A centralized MS cannot scale for large sizes; thereby, it requires decentralization. In [1], we describe a scalable, hierarchical, distributed MS called SIMONE.

Most solutions for a high-performance MS have one common characteristic: they are static, that is, configured manually at start-up time based on the prevalent conditions. The dynamic nature of the NCS environment has been largely overlooked (some exceptions are [2], [3] and [4]). This dynamic nature manifests itself in two ways: (i) changing operating conditions of the computing system, e.g. network load, processor load, varying number of nodes in the NCS pool altering resources available for monitoring tasks and the number of nodes to be monitored, and (ii) changing monitoring requirements, e.g. when parameters need to be updated more frequently. Monitoring requirements may be determined by the application, user, or the system.

Due to the dynamic nature of the NCS and changing monitoring requirements, a distributed configuration initially designed to be optimal may perform poorly over extended periods of time. Besides MS performance, the monitoring overheads on NCS may increase due to the skewed usage of resources by the MS. We propose a reconfigurable MS as a solution to the above performance problem.

Reconfiguration is an adaptation technique whereby the components of the MS, or the (logical) connections among them, change automatically, adapting themselves to the changing environment with the aim of fulfilling monitoring requirements and reducing overheads. Additional reasons for reconfiguration could include administrative demands, node failures,

freeing nodes that are needed for running critical applications.

In this paper we propose and discuss a reconfiguration mechanism that can be incorporated in SIMONE. A distributed MS has managerial tasks (operations performed by the manager) spread across several nodes which are also running NCS jobs. We propose a metric called temperature which summarizes the performance of a node. Depending on the temperature value, a merge, split or migrate transformation may be performed on the distributed manager entity, resulting in the reassignment of managerial tasks or responsibilities (measured in terms of number of managed nodes).

A reconfigurable solution must be simple and efficient (imposing low overheads), scalable (we anticipate its use in large-scale NCS), and capable of integration with existing monitoring methods. Other important issues are flexibility (the capability of dealing with situations not foreseen at design time), fault-tolerance (no single point of failure and capability of reacting to faults), adaptability (quick response to changes in requirements), timeliness (quick transformations), seamlessness (transparency to users), and stability (ability to function even in the worst possible conditions).

The complexity of the computing environment and interactions between NCS and MS precludes an analytical approach to assess the effectiveness of our proposal. We therefore use a queuing model to capture the most important interactions and use simulation to conduct performance studies.

The remainder of the paper is organized as follows. Section 2 provides some architectural background. Section 3 models the NCS, MS components, and applications running on the NCS, and describes metrics for assessing MS performance. The reconfiguration system, issues, and implementation are discussed in Sections 4 and 5. Section 6 describes the simulations conducted and results obtained. Related work is discussed in Section 7, and conclusions in Section 8.

2 Towards a Reconfigurable Architecture

Manager and agent are the two main entities in an SNMP-based MS which interact using the client/server model. A manager (the client role) requests and obtains monitoring information from several agents (the server role), while the agents listen and respond to manager requests

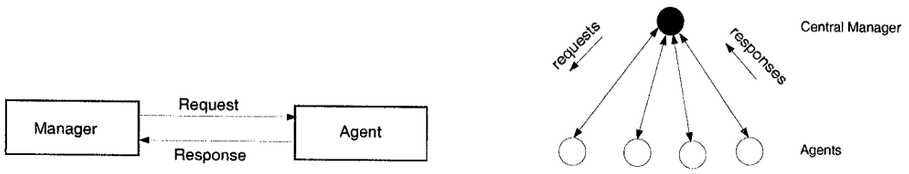

Figure 1: Model of a centralized MS using the client/server paradigm

(see Figure 1). In a centralized set-up, the single manager becomes a bottleneck for agents numbering more than 100 to 200 (see [1]). A distributed MS, SIMONE, was proposed as a solution. Several intermediate-level managers (ILMs), in one or more levels, are delegated managerial tasks by the top-level manager (TLM) which, however, retains the overall MS control. An ILM is a dual entity: an agent to its manager (a TLM or an upper-level ILM) and a manager to its agents (see Figure 2). Its duties are defined as downloadable scripts, providing flexibility, while communication continues to be strictly SNMP.

While the distributed SIMONE reduces monitoring latency by an order of 2-10 [1], its configuration is static; the monitoring tree is defined at startup time, thus limiting performance improvement. A mechanism to change dynamically the number of ILMs, and the topology of the tree dynamically by reconfiguration is now proposed and will be described further in Section 4.

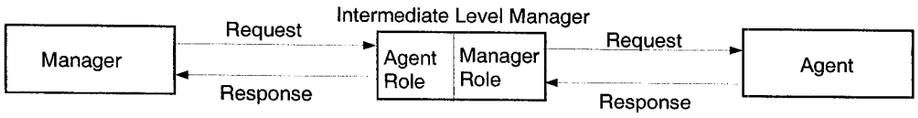

Figure 2: Model of a distributed MS with the inclusion of a level of ILMs

3 Model of the MS in an NCS Environment

An NCS is a collection of computing nodes (resources) in which applications (collections of jobs/tasks) request and gain access to some resources for an interval of time. They are broadly divided into NCS and MS jobs. The following assumptions are made

1. NCS jobs are assigned to nodes by an independent scheduler and are beyond MS control. Each node has its own scheduler. Efficient use of the NCS is the responsibility of the

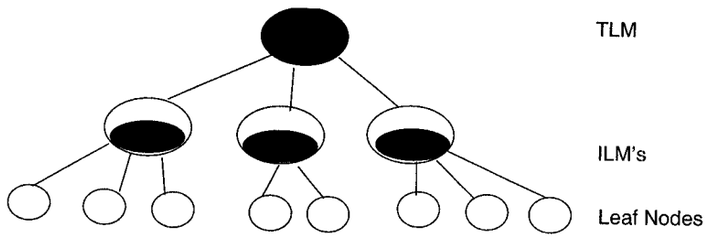

Figure 3: Structure of a monitoring tree with one level of ILMs. The shaded area denotes a manager or an entity performing the role of a manager. Non-shaded areas denote agents or the agent role.

NCS scheduler and efficient node usage is the responsibility of the node scheduler.

2. The MS, an application by itself, also shares (rather competes) for NCS resources; individual resource usage by the MS is determined by the local scheduler of that resource.

3. MS jobs have lower priority than NCS jobs.

4. An NCS application is modeled as a collection of NCS jobs that first complete computation followed by communication.

3.1 Node and Job Classification

To aid the model description, we classify the computing nodes and jobs into different categories based on their role in monitoring. Depending on the nature of the jobs processed, nodes can be classified into three classes,

- Leaf: The predominant and default class of monitored nodes, which do not perform any monitoring task other than running an agent.

- TLM: A single node dedicated to performing managerial tasks and responsible for the overall control of the MS. It does not process NCS jobs.

- ILM: A collection of selected nodes also perform managerial tasks and are called ILMs. An ILM, being a dual entity, behaves as a leaf as well as a TLM. The priority of both the leaf and ILM nodes is executing NCS jobs.

The job classes are,

- MS-Agent: Models the agent tasks. The agent periodically updates variable values in a virtual table called the MIB with monitoring data gathered from node's kernel.

- MS-Poll: Models those tasks performed due to a manager poll. Poll is a periodic query from the manager to the agent requesting MIB variable values.

- MS-Report: Models the tasks derived from the reception of a SNMP report from a leaf or a lower-level ILM. A report is a monitoring summary information representing a set of nodes and is unique to the distributed architecture. Reports are consolidated and processed in the ILM to generate a single output report which is forwarded to the next higher level node.

- NCS jobs: Models actual jobs submitted by the NCS scheduler.

Jobs are characterized by the size or magnitude (time units to execute a job) and inter-arrival time (time between jobs of the same class). The interactions (Figure 4) between different nodes and job classes are summarized as follows. A leaf node executes different NCS, MS-poll and MS-agent jobs. The MS jobs forwarded to an ILM are MS-report jobs. An ILM executes all classes of jobs executed by a leaf node and, additionally, the MS-report jobs received from the leaves or lower-level ILMs. The TLM receives and executes MS-report jobs besides its local MS-agent jobs. However, being a dedicated machine for monitoring, it does not execute any NCS jobs. As a final consumer of monitoring information, it does not generate any monitoring output except when requested by an external user/application. This interaction is outside the scope of the model and therefore not considered.

3.2 Modeling the Nodes

Node classes can be described with a queuing model (see Figure 4). All nodes have a CPUQ, representing the CPU. It is a processor-sharing server with an infinite queue length, and a service time proportional to the magnitude of the job. All classes of jobs pass through the CPUQ. Leaves also have an OCommQ, a FIFO queue with fixed service time and an infinite queue length, that represent the output communication interface. The TLM does not have an output interface, because it does not forward monitoring data. However, it has input interface, ICommQ, since it receives reports from ILMs. Its characteristics are

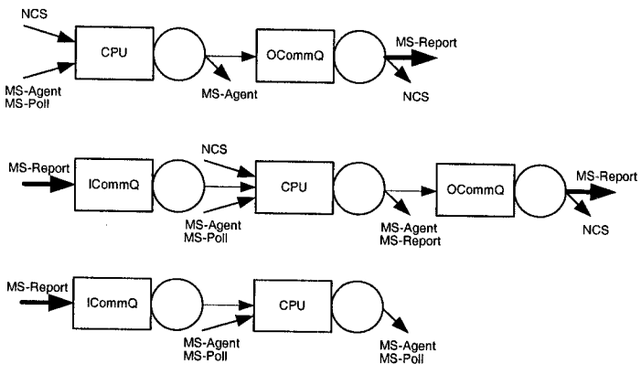

Figure 4: Models of the components of the monitored NCS: leaf (top), ILM (middle), TLM (bottom). TLM does not process NCS jobs.

similar to OCommQ. An ILM has both ICommQ and OCommQ queues to represent both communication interfaces. Being a dual entity, it combines the elements of a leaf node with those of the TLM.

Previous experiments reported in [1], [5] show that, under normal conditions, the communication network is not the bottleneck in the monitoring activity (most networks operate well below their peak capacities). For this reason, we focus on CPU usage and model just certain parts of communication activity. Reception of NCS and poll jobs is not considered. Hence, we model only responses and not requests. Instead, local generators are used in the model. This way simulation set-ups are simpler, while maintaining accuracy when modeling CPU and OCommQ. However, the reception of a poll triggers CPU usage. Hence, we model only responses and not requests. Also, while computing temperature values, our emphasis is on load (representing CPU usage), rather than on latency.

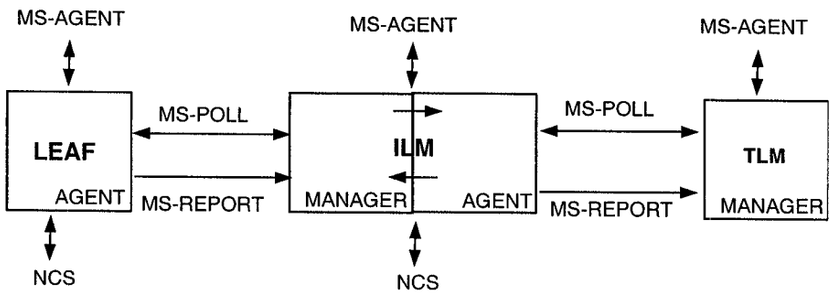

Figure 5: Combining jobs and node classes. Note, that we model job flow only in the bottom-up direction, i.e. leaf to TLM.

3.3 Input and Output Characterization of Different Classes of Jobs

The characteristics of different job classes is discussed below. The agent in all monitored nodes updates a fixed number of variables in its MIB table at regular intervals of time given by Agent Update Period (AUP). The job size and inter-arrival times are thus constants.

Higher-level managers periodically poll agents of lower-level nodes for a fixed number of MIB variables. The period is given by Manager Polling Period (MPP), an MS operating parameter. Like MS-agent jobs, MS-poll jobs have fixed magnitude and constant inter-arrival time given by MPP.

MS-Report jobs are a consolidation of monitoring information at different node levels. The output of a MS-Poll job at one node triggers the generation of a MS-Report at the input of the next higher-level node. Since MS-Poll experiences node delays, the inter-arrival characteristics of MS-Report jobs cannot be determined *a priori*.

NCS jobs are random magnitude jobs at random inter-arrival times. The probability density function of these parameters have been determined from the actual data gathered from a real NCS system, PUNCH [6], developed and deployed at Purdue University. Job parameters were obtained using a distribution-fitting program [7] on about 40,000 entries from PUNCH logs. The best fit for magnitude distribution is a Pareto type II [8], [9], with parameters (0.0667, 0.3948) shifted 0.02; for inter-arrival time distribution it is a Pearson type V [8] [9], with parameters (1.1009, 62.613) shifted -10.42 (units are in seconds). For simplicity in the simulations, we used exponential distributions for both job size 50.286 (shift 0.0189) and inter-arrival time 470.44 (shift -0.0103). These distributions were found to be reasonable approximations of the best-fit distributions.

Thus, the magnitude of all MS jobs are fixed, the inter-arrival times of MS-agent and MS-poll jobs are constant while MS-report jobs are dependent on node delays. The distribution characteristics and values of inter-arrival time and job magnitude form inputs to signal generators used in the simulations. These values for MS-agent and MS-poll jobs are obtained from previous experiments [1], [5].

3.4 Complete System Modeling

The missing link needed to complete this model is the network. Since the focus of the model has been on node performance and not on the network, a simple model of the network with an all-to-all connectivity with fixed communication delays is sufficient. Results from previous experiments [1] showed small network intrusion partly because NCS are usually connected via high-performance networks, operating well below peak capacity.

3.5 Performance Metrics

Metrics are time varying functions that characterize the performance of an entity in a system, application, or a resource. Common MS performance metrics are:

- CPU intrusion: overheads imposed on the nodes due to the execution of monitoring tasks,
- Network intrusion: overheads imposed on the network due to the exchange of monitoring information, and
- Latency: an end-to-end metric measuring time since a manager polls a node, until a response is received.

Latency is redefined in this paper to mean node latency, i.e., the time interval from when a MS-poll job enters a node until it leaves the node (or the last queue). Latency and intrusion indicate the level of MS activity and performance on a node.

The two global performance objectives of the reconfiguration system proposed in this paper are: maximizing MS performance and minimizing MS intrusion (impact of MS on NCS applications). We now introduce a new metric called temperature in the reconfiguration system. **Temperature** of a node is a local metric, which assesses the performance of an ILM node and the impact of monitoring tasks on NCS applications. It is expressed as a weighted average of MS jobs' node latency, CPU and communication overheads of the node.

$T = aI+bL$

where,"I" is CPU intrusion and "L" is (node) latency, and "a" and "b" are weight factors.

Temperature values above or below certain thresholds will trigger reconfiguration operations or transformations. The objective of the mechanism is reflected by the values of "a"

and "b"; "b" should be large for minimizing node latency while "a" should be large when reduction of intrusion is the goal. Note that intrusion has a direct effect on latency (a heavily used node takes longer to process jobs). Since the load is subsumed in latency value, we currently assign a weight 1 to latency and 0 to intrusion. The administrator chooses and changes weights, and sets temperature threshold values in accordance with user requirements. Both can be changed dynamically. The temperature expression can also be changed to reflect additional or different input factors.

We need a global metric to evaluate the performance goals of an entire system and to compare different MS alternatives (static and reconfigurable). We define a global metric called global node latency. Two other local metrics are the average and the standard deviation of ILM node latency. These three metrics are defined below.

- Average node latency: The mean delay experienced by poll jobs in a particular ILM. Thus, each ILM has an average node latency.
- Standard deviation of node latency: Standard deviation of delays experienced by poll jobs in a particular ILM. Again, each ILM has a standard deviation of node latency.
- Global node latency: mean value of average node latencies of all ILM nodes. A single value for the whole MS.

A smaller global node latency indicates an overall lower impact on the NCS applications. The average ILM node latency provides the same information, but for a single node. Standard deviation is measured for a single ILM node. It measures the variations in latency experienced by poll jobs in the ILM node over a period of time.

4 Reconfiguration

This section defines temperature and transformations and describes how temperature is computed.

4.1 Temperature and Transformations

Mechanisms to reduce the overall temperature of the system fall into two categories,

1. Reducing the activity of the monitoring system, e.g. increasing AUP and/or MPP, to reduce the number of parameters measured.

2. Modifying architectural characteristics of the MS to improve its performance without changing MS operational parameters.

Both the approaches are complementary and can be combined. Currently, we assume that the MS operational parameters are already optimally adjusted to satisfy application requirements and further changes compromise monitoring requirements. In this paper we focus only on the second approach.

MS performance can be improved beyond decentralization [1] if the ILM configuration can be allowed to change. The arrangement of (logical) connections from leaves to ILMs can be dynamically modified to improve efficiency by moving managerial MS tasks from heavily loaded ILMs to less loaded ILMs. We call this reconfiguration.

The transformations to be applied on ILMs is decided based on current values of temperature which are periodically computed. Keeping the large size of NCS in mind, a localized decision process involving two adjacent levels of nodes is used (see Figure 6). The TLM and ILMs maintain a list of nodes they manage called the supported node list. All ILMs also compute and store their local temperature. During a poll, a node at level I obtains the temperature and supported node list of all $I+1$ ILMs it manages. Additionally, the level I node has a list of unused ILMs within the same network region. The level I node computes transformations for all its managed $I+1$ ILMs and reassigns the level $I+2$ nodes. In the final step, the level I node updates the supported node list of the $I+1$ ILMs. Thus, the decisions for level $I+1$ ILMs are made by the next higher level I. In the special case involving a single level of ILMs, the TLM (now level I) is the only node which determines the transformations that are enacted on the single $(I+1)$ level ILMs. The transformations are now described as,

- Migrate- a node stops running an ILM and its managerial tasks are assigned to another, less "hot" node.

- Split- a node does not stop running an ILM but decides to pass part of its managerial tasks to another node.

- Merge- the complement of split. A "cold" ILM assumes the managerial tasks of another cold ILM, which then becomes a regular node.

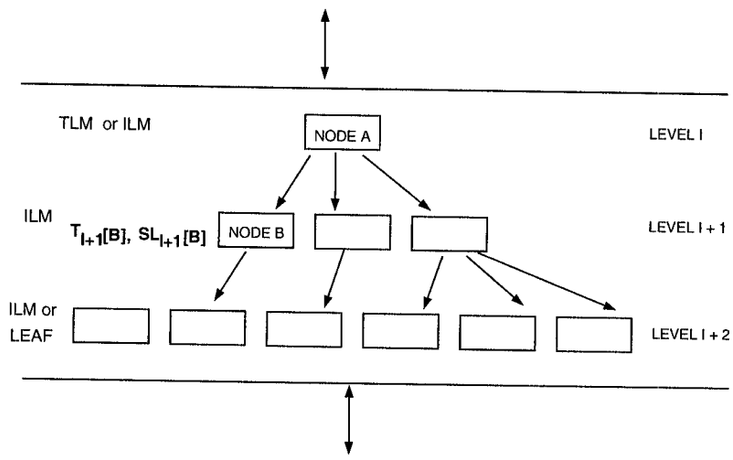

Figure 6: Segment of the distributed MS showing level I, $I+1$ and $I+2$. Each ILM stores temperature, T, and a list of its supported nodes, SL. Transformations for ILMs at level $I+1$ are decided by level I nodes.

Migrate and split operations try to reduce the temperature of an ILM. The utility of the merge operation when two nodes are operating within allowable ranges may be debated. The reasons are to free a lightly loaded node that can be fully dedicated to process NCS jobs, experiments show that due to additional communication and synchronization costs, too many ILMs are counterproductive to performance [1], and for symmetry, just as new resources are added when required (split), resources should be freed when not required.

4.2 Issues and Policies

A reconfiguration solution needs, necessarily, to be simple. The cost of the mechanism would otherwise be higher than the performance gains. Both, the choice of transformations and the reconfiguration mechanisms are therefore kept simple in our proposal.

The reconfiguration operations are integrated with existing monitoring process. Extending the MIB and defining additional MIB variables meet storage requirements. Reconfiguration operations are performed by scripts similar to those used for other monitoring operations executed by ILM (see [1]). Both scripts are executed during the periodic polls.

The algorithm continually improves the performance over a succession of simple, localized transformations, instead of having a single global element computing the optimal configuration for the entire MS at each step. Global decision processing allows for a better decision-

making, but relies on complete accurate information. Accurate information needs frequent propagation over the entire system, making this option not scalable and therefore infeasible. Additionally, a central authority is a single point of failure.

An alternative to using polls to trigger reconfiguration, is to use triggers: ILMs monitor their temperature, and those of their neighbors, and decide to (partially) change their configuration whenever a certain threshold is reached. Triggers are more useful to deal with large performance spikes that need prompt reconfiguration. Polling has lower overheads, enabling us to use shorter polls and reduce performance degradation over sustained periods, which we see as a larger concern than spikes. Polling is also simpler to integrate with SIMONE. Polling period needs to be properly adjusted to avoid dealing with stale data.

An issue not yet explicitly considered is that of fault tolerance. However, the localized decision process and the migrate transformation, together, may be explored as a means to achieve fault tolerance.

5 Implementation

The proposed reconfigurable MS will be implemented within the framework of SIMONE, while preserving two design goals of SIMONE: the role of manager and agent, as defined by SNMP, and adapting SNMP primitives to perform all required operations within SNMP.

The advantages of these design goals are the resulting tool continues to be compatible with other SNMP systems, and it avoids proprietary solutions that would make integration with other monitoring systems difficult.

In order to implement a reconfigurable system, the following mechanisms need to be implemented:

- A way to record local variables to compute the temperature at each node.
- A way to compute and store temperature.
- A way to trigger a reconfiguration operation.
- A way to perform the corresponding transformation.
- A way to assess the cost-benefit of performing a reconfiguration, avoiding loops and costly operations.

An outline of the implementation of the reconfigurable MS is presented here. The basic extension consists of using an extended MIB to store information needed for reconfiguration

and using SNMP communication primitives to retrieve this information as well as to trigger architectural transformations.

Each ILM has an MIB table to store monitoring variables. The table is extended to store additional variables needed to compute temperature. Additionally, MIB variables are assigned to store the computed temperature (temperature-result) and the supported node list.

We assume that there is a set of nodes which are able to additionally perform as ILMs. Active ILMs are those nodes that are actually performing managerial operations, while inactive ILMs are those that are discharged of managerial responsibilities (but can accept them if required). The TLM stores both these lists as MIB variables as well.

Since we consider only one level of ILMs at present, transformation decisions are made by the TLM. The algorithm works for multiple levels of ILMs as well (see Section 4). A TLM polls the ILMs and receives monitoring data, with which it computes monitoring parameters. In the reconfigurable mode, the TLM also polls the temperature of the ILMs and stores the status.

Using the temperature of ILMs, the TLM decides the the need for reconfiguration and computes the new configuration, reassigning responsibilities among the affected ILMs. This reassignment includes halving the list of dependent nodes (split), doubling it (merge), making it null (migration) in one ILM, and the reassigning the list of nodes to the other node in migration. Updated node lists are sent to the affected ILMs (via a SNMP set primitive). The TLM also performs the required bookkeeping operations, such as maintaining the lists of active and inactive ILMs.

Current design restricts transformations to inter-level ILMs, while the number of levels of ILMs remains constant. New transformations called *fork* and *join* are proposed for the future and could add or reduce the number of levels of ILMs. A fork operation creates two children below it. Symmetrically, a join affects at least three nodes, a parent and two or more children, collapsing to become a single node.

6 Simulation and Results

Using the Ptolemy [10] environment as a simulation engine for the model described in Section 3, we have conducted a set of simulations to evaluate the performance of the proposed reconfigurable system. The simulation set-up uses off-the-shelf components provided by Ptolemy (such as FIFO queues, signal and trigger generators, random number generators), complemented with others designed by us specifically for our model (information stamping devices, data modifiers, gates, recorders, statistical modules, processor sharing server, etc). Collectively, these components form three classes of modules: input generators (to inject NCS, MS-Agent and MS-Poll jobs, and to trigger MS-Report jobs), nodes (leaves, ILMs and TLM), and the routing module described below.

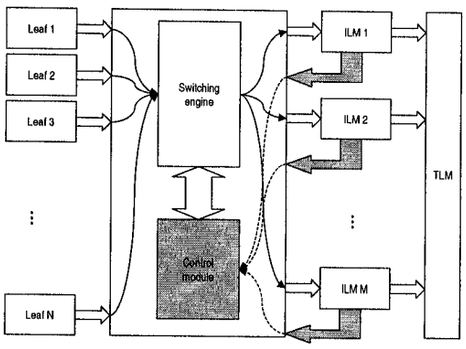

Figure 7: The routing module. The shaded and dotted areas show the temperature feedback.

6.1 The Routing Module

For static configurations, the output of a node can be directly connected to the input of another node. In the reconfigurable scheme, the logical connections from leaf nodes to ILMs are constantly changing due to reconfiguration operations. In order to simulate this reassignment, a routing module is placed between leaves and ILMs, as shown in Figure 7. This module is a simulation artifact with no counterpart in the actual system and is used to simulate the reconfiguration logic. It has N input ports (one per leaf), M output ports (one per ILM) and M control ports. These control ports receive temperature values from the ILMs. Based on the temperature value, the routing module modifies an internal routing

table that allows traffic to flow between a leaf and the ILM to which it reports. An internal set of data structures allows bookkeeping tasks such as maintaining the list of unused ILMs. Multiple levels of ILM would require using several routing modules. Although our proposed reconfigurable system allows any number of levels of ILMs, the performance evaluation has focused primarily on single level ILMs. All the discussions below pertain to single level ILMs, unless mentioned otherwise.

6.2 Measurements

Simulation set-ups were designed for three different MS schemes: Centralized, Distributed and Reconfigurable. The MS configuration was varied to contain 64-2048 leaf nodes with 2-64 ILMs. In the reconfigurable scheme, the initial number of active ILMs was set at a number lower (we happen to choose half) than the total number of ILM-capable nodes, to allow for expansion over time. The job parameter (inter-arrival time and magnitude) values were obtained from previous experiments as described in Section 3.3. These jobs are referred to as NCS-P(PUNCH) jobs. Another set of NCS jobs, called NCS-H jobs, which are smaller in size were also used (parameter values of average job size and inter-arrival time are, 20 and 5 seconds, respectively). These are more frequent and cause higher loads. Simulations were conducted for a single level of ILMs. Time units are defined as simulated time units (stu), where each stu is a clock tick in the simulation and set to be equal to 100 ms of real time. All the simulations were run for the same number of stu's. Statistical components designed by us were used to record the timing data by observing jobs at various stages of the model.

The centralized and distributed schemes were used for ensuring the validity of the simulation set-up by comparing simulation data with those obtained from previous experiments ([1], [5]), performed on actual testbeds. Simulation experiments were designed to

- Compare the global and local ILM latency for different static and reconfigurable schemes for 64-2048 leaves with a single level of 2-64 ILMs. The simulations were conducted for both NCS-P and NCS-H jobs.

- Explore the impact of thresholds on MS performance. Thresholds were varied for the reconfigurable MS to allow one of the four transformations to dominate.

- Record the number of transformations occurring in each ILM for a period of time, the duration of time the ILM is in the "hot zone", the average temperature of the ILM while it is in the "hot zone", and the performance gain per transformation. This work is still underway.

6.3 Results and Interpretation

Simulations were conducted for the above list, and the values of metrics defined in Section 3.5 were observed. The results for sample configurations are presented below. Table 1 compares global latency for distributed and reconfigurable schemes for 512-2048 nodes and 4-64 ILMs. The reconfigurable scheme has smaller global latencies in the region where the performance of static distributed scheme starts to deteriorate. This occurs when the manager node supports in excess of 100 nodes [5], [1], causing the manager to become a bottleneck.

Figure 8 compares the average ILM latency and deviation of latencies for ILM nodes for static distributed and reconfigurable schemes for a sample configuration of 1024 nodes with 8 ILMs. The standard deviation for an ILM is the variation of latencies for that ILM over the duration of time the simulations were run. The average delays for the reconfigurable scheme were lower, with less variations. The standard deviations were also lower.

Global latencies were also measured for simulations using the NCS-H jobs (instead of NCS-P). Global latencies using NCS-H and NCS-P for 1024 node static distributed schemes are compared in Table 2. The increase in latency can be attributed to higher loads imposed by NCS-H jobs.

The threshold values used in the simulations were [0-1.45-10-20-Max], where temperature between 0 and 1.45 stu triggers merge, 2 and 10, causes no change, 10 and 20, a migrate transformation and values above 20, a split. A set of different threshold values were selected whereby, in each case, one of the transformations occurs over a larger temperature range (a wider spread between thresholds, is one reason for that transformation to occur more frequently). Global latencies for different threshold sets for the reconfigurable scheme with 1024 nodes, 8 ILMs and using NCS-H jobs, are given in Table 3. The lowest value is observed in the merge case. This is because, with only 8 ILMs in the pool, freeing of ILMs allows other ILMs to migrate or split.

Approaches to threshold selection in our view can be based on user requirements or based

achieving optimality. Since we are guided by user requirements, we allow the administrator to choose thresholds (possibly aided by translation of requirements, such as permissible delay, to threshold values). Mechanisms for automated threshold computations are feasible; currently we consider these enhancements as an open line of future work.

Simulations to measure the number of transformations occurring in each ILM for a period of time, the duration of time the ILM is in the "hot zone", the average temperature of the ILM while it is in the hot zone and the performance gain per transformation are underway.

Number of ILMs	512 SD	512 RC	1024 SD	1024 RC	2048 SD	2048 RC
4	3.35	1.76	4008.36	3586.00	7172.00	4368.00
8	1.53	1.56	3.42	1.58	4006.00	3568.00
16	1.468	1.56	1.73	1.5	3.19	2.53
32	1.512	1.49	1.49	1.48	1.9	1.7
64	1.46	1.46	1.46	1.49	1.5	1.49

Table 1: Comparison of global latencies for distributed and reconfigurable schemes for 512-2048 NCS sizes, with 4-64 ILM's using NCS-P jobs. "SD" has been used above to abbreviate static distribution, while "RC" has been used for reconfigurable scheme. Saturation occurs when the ILM has to manage a large number of nodes. Units are in stu.

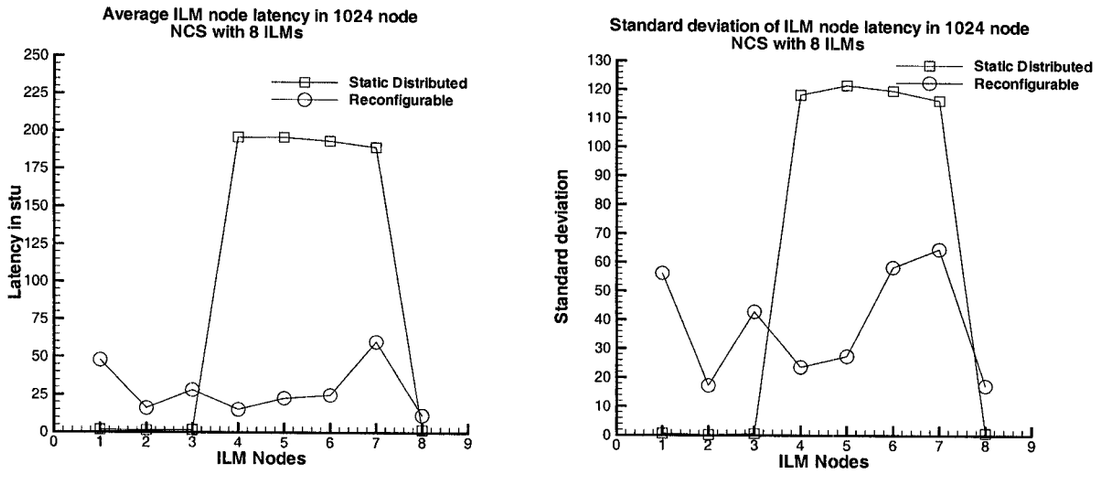

Figure 8: Comparing average latencies and deviation of latencies in ILM for distributed and reconfigurable schemes for 1024 nodes/8 ILMs using NCS-H jobs. An stu equals 100 ms.

Number of ILMs	NCS-P jobs	NCS-H jobs
4	4008.36	4036.77
8	3.42	109.47
16	1.73	2.0302
32	1.49	1.746
64	1.46	1.7

Table 2: Comparison of global latencies for distributed scheme for NCS-P and NCS-H jobs for 1024 nodes. Units are in stu.

Threshold Set	**Transformation with increased threshold range**	**Global latency (stu)**
[0-5-10-20-5000]	Merge	15.04
[0-2-15-20-5000]	No Change	33.03
[0-2-10-50-5000]	Migrate	149
[0-2-10-15-5000]	Split	40.66
[0-2-3-5-5000]	(Narrow spread)	35.36
[0-1.5-10-20-5000]	Values used in the simulation	28.96

Table 3: Effect of thresholds for 1024-8 reconfigurable configuration, using NCS-H jobs.

6.4 Discussion on Cost of Reconfiguration

The cost benefit tradeoff between reconfiguration and overheads (or cost) needs to be weighed in before enacting a reconfiguration. Both, the decision process and the transformation enactment steps contribute to the overheads. Overheads are measured in terms of complexity of the algorithm (computational costs and memory), storage costs, communication costs and delays.

In our reconfiguration system, the algorithm computes decisions locally depending on the threshold values defined, and temperature values, and computes the new node list from the existing one. If ILM at level i supports X ILMs, the overheads for ILM level i are: computational complexity of decisions is O(X), storage is $(X+C)$ MIB variables, and communication: $2X$ SNMP requests to send and receive node lists. The costs of reconfiguration has not been fully explored, further work needs to be done both in expressing costs of reconfiguration and incorporating it in the decision making process of the reconfiguration scheme.

7 Related Work

Tools designed for network computing environment are: NWS [11] which provides dynamic resource performance forecasts; Globus' Gloperf [12] tool which works like NWS, and periodically schedules end-to-end tests to retrieve latency and bandwidth information; Netlogger [13], a trace-based system used mainly for diagnosis, and Remos [14] which combines different monitoring techniques such as SNMP and end-to-end tests focusing on providing information for network-aware applications. Performance issues when working in large networks are tackled by using a hierarchy of groups in NWS and Globperf, by stopping and starting logging process in Netlogger, and by implementing a hierarchical monitoring architecture using a data collector in Remos. These solutions differ from ours because they use non-standard communication protocols (while SNMP is seen as an enhancement) and more importantly, their configuration is static.

The Grid Performance Working Group of the Grid Forum has ongoing work towards defining a monitoring service architecture for the Grid [15]. SIMONE will be made compliant with these anticipated framework specifications to allow it to interact with other Grid systems.

While the concepts of reconfiguration, migration etc., have been widely developed and studied in other areas [16], most monitoring systems have been designed with a static configuration. The delegation paradigm [17] introduced in SNMP allowed dynamic runtime delegation of monitoring tasks assigned by the administrator manually, without feedback information. Mobile code paradigms for network management have suggested taking advantage of mobile code mechanisms such as migration. Liotta [16] uses a dynamic hierarchical manager model based on delegation using mobile agents, for scalable monitoring. Optimal location of migration intelligence, and minimization of network traffic and delay is sought by using migration and cloning transformations. Liotta indicates some of the complexities and drawbacks of using mobile code. Our solution in contrast uses SNMP mechanisms alone, and is generalized to be applicable to any SNMP-based MS. Lutfiyya's [2] adaptive model, studies the reconfiguration of monitoring elements in response to dynamic changes in management requirements, user/system constraints and resource availability. In a testbed using Common Management Information Protocol [18] (CMIP) in Open System Interconnection [18] (CMIP) (OSI) framework, an optimal solution to minimize monitoring overheads is sought. Lutfiyya

states the need for his solution to be examined for larger systems. Our algorithm trades the complexity and high overheads of an optimal solution for a simple approximate solution which is better suited for larger computing systems. Hollingsworth [4] proposes an adaptive cost system for Paradyn performance tools in order to maintain predefined levels of permissible perturbations in applications by delaying dynamic instrumentation until execution. This allows the flexibility of dynamic insertion and alteration. Performance is gained by reducing instrumentation at the cost of monitoring accuracy. Although specific to application monitoring, the solution can be viewed as an adaptive solution without reconfiguration. Our solution assumes that polling frequency cannot be reduced further without compromising monitoring accuracy requirements.

The Quorum [19] program by DARPA is developing global distributed computing capabilities for mission-critical applications with the fundamental premise that QoS management holds the key. One of the initiatives is an adaptive global resource management to dynamically discover, allocate and schedule resources from a global pool to an application based on QoS requirements. Some of the groups ([14], [20], [21], [22]) are working on monitoring or management systems for large Grid-like computing systems. Efforts have also been on to investigate performance of streaming data to visualization using transformations similar to ours. Our goals differ in the sense that reconfiguration seeks to improve the monitoring application performance.

8 Conclusions

This paper proposes an adaptive mechanism to enable an MS to react to changes in operating conditions in order to minimize monitoring overhead impact on NCS applications, and maximize MS performance. A reconfiguration algorithm is proposed, which, based on current values of a local node performance metric called temperature, decides whether to enact a transformation (merge, split, migrate) affecting two ILMs.

The proposed mechanism can be implemented on SIMONE, a distributed SNMP-based monitoring system. Its implementation envisages the use of new MIB variables and additional scripts for the ILMs. Communication is always performed using SNMP primitives; this includes the triggering of transformations, which are synchronized with the periodic

monitoring polls.

A queuing model is used to describe MS in an NCS environment and study the interactions among different classes of jobs. The Ptolemy simulator environment is used to design a number of components to simulate and evaluate this model. Simulations were performed for different configurations, 1-2048 nodes and 2-64 single level ILMs by using data from previous experiments to set input generator parameters. Simulations for shorter NCS jobs and different sets of thresholds were performed. Preliminary results indicated that reconfiguration has lower global ILM node latency (over time), lower average ILM node latency, and lower latency deviation for ILMs. Reconfiguration performs better than static distribution in the region near saturation. In this region, the static distribution performance starts to deteriorate.

Further investigation with simulation data, computing performance gain per transformation, convergence and work on reconfiguration costs is proposed. Future enhancements under consideration are adding fork and join transformations and exploring the feasibility of automating threshold value selection.

References

[1] Rajesh Subramanyan, José Miguel-Alonso and José A.B Fortes, "A Scalable SNMP-based Distributed Monitoring System for Heterogeneous Network Computing", *in Supercomputing*, Nov. 2000.

[2] Hasina Abdu, Hanan Lutfiyya and Michael A. Bauer, "A Testbed for Optimizing the Monitoring of Distributed Systems", *in Proceeding of PDCS '98*, 1998.

[3] Antonio Liotta, George Pavlou, Graham Knight, "A Self-adaptable Agent System for Efficient Information Gathering.", *in MATA*, pp. 139–152, 2001.

[4] Jeffrey K. Hollingsworth and Barton P. Miller, "An Adaptive Cost System for Parallel Program Instrumentation", *in Euro-Par '96*, Aug. 1996.

[5] 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 Univ., July 2000.

[6] Nirav H. Kapadia and José A. B. Fortes, "PUNCH: An Architecture for Web-Enabled Wide-Area Network-Computing", *Cluster Computing*, vol. 1, pp. 119–131, Sept. 1999.

[7] *Palisade Software Developers, Newfield, NY.*, http://www.palisade.com/html/bestfit.html.

[8] Richard Saucier, *Computer Generation of Statistical Distribution*, Army Research Laboratories, http://ftp.arl.mil/random/random.pdf, March 2000.

[9] Brian Everitt, *The Cambridge Dictionary of Statistics*, Cambridge University Press, 1998.

[10] *Ptolemy 0.7 User's Manual, University of California, Berkeley, http://ptolemy.eecs.berkeley.edu.*

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

[12] 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.

[13] 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.

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

[15] I. Foster, C. Kesselman, S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations.", *in Intl. J. Supercomputer Applications*, 2001.

[16] Antonio Liotta, Graham Knight and George Pavlou, "On the Performance and Scalability of Decentralized Monitoring using Mobile Agents", *in DSOM*, 1999.

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

[18] U. Black, *Network Management Standards*, McGraw-Hill, 1995.

[19] DARPA, *Quorum Project*, http://www.darpa.mil/ito/research/quorum/index.html.

[20] Andrew Chien, Klara Nahrstedt, and Jane W.S. Liu, *EPIQ, End-to-End Quality of Service and Resource Management*, http://cairo.cs.uiuc.edu/epiq/index.htm#Projects.

[21] Kane Kim, *Adaptive Resource Management in RT OO Global Information Systems*, University of California at Irvine, http://dream.eng.uci.edu/darpa9701/.

[22] Yair Amir, Baruch Awerbuch, R Sean Borgstrom, et. al, *Research Management for Metacomputers*, Johns Hopkins, http://www.cnds.jhu.edu/research/metacomputing/.

[23] Daniela Rosu, Karsten Schwan and Sudhakar Yalamanchili, "On Adaptive Resource Allocation for Complex Real-Time Applications", *in Proceeding of Infocom 2000*, 2000.

[24] A. Waheed, D.T. Rover, M.W. Mutka, H. Smith, and A. Bakic, "Modeling, Evaluation, and Adaptive Control of an Instrumentation System", *in Proc. Real-Time Technology and Applications Symp. (RTAS '97)*, June 1997.

[25] Manfred Siegl, *Design and Realization of a Mid-Level Management System*, PhD thesis, TUW-HDNM, Vienna University of Technology, Nov. 1996.

[26] Aiko Pras, *Network Management Architectures*, PhD thesis, Centre for Telematics and Information Technology, University of Twente, April 1995.