DYNAMIC LOAD BALANCING FOR DISTRIBUTED NETWORK MANAGEMENT

Kiyohito Yoshihara

Manabu Isomura

Hiroki Horiuchi
KDDI R&D Laboratories Inc.,
2-1-15 Ohara Kamifukuoka-shi
Saitama 356-8502, Japan
yosshy@kddilabs.jp, isomura@kddilabs.jp, hr-horiuchi@kddilabs.jp

Abstract: The scalability limitations of centralized management models have motivated distributed management models, in which management programs describing some of management tasks are distributed and executed on managed systems. In the models, management program distribution that considers dynamic network resource utilization is one of the most important challenges, in striking a load balance between management and managed systems for an entire managed network. Some methods for load balancing have been studied; however, they cannot adequately be achieved throughout an entire managed network. This arises from criteria for load balancing that lacks dynamic network resource utilization, or from a localized subnetwork in which the performance is limited, although it does include processing loads for dynamic network resource utilization. To solve this, a new dynamic load balancing method is proposed for distributed network management. Thus, systems that execute management programs are decided dynamically on the basis of CPU utilization for each system and the bandwidth required for executing all management programs. Two typical algorithms derived from the proposed method, each having different criteria in the form of mean deviation and range types with respect to CPU utilization, are introduced. They were evaluated analytically according to capability, i.e., how well they perform as close to load balancing as possible, as well as time complexity. The results show that the mean deviation type algorithm performs better at almost the same computational cost. A prototype system is also implemented based on the proposed method, and evaluated empirically by applying it to an operational LAN. The proposed method performs well in trials with a trivial overhead.

Keywords: Distributed network management, load balancing, management by delegation, mobile agent, SNMP

1. Introduction

Centralized management models, such as SNMP (Simple Network Management Protocol) and TMN (Telecommunications Management Network), address scalability limitations, since management tasks, including management information-gathering, information analysis and results-oriented system controls are focused on a centralized management system, as well as increasing numbers of managed systems resulting in management traffic overhead [13, 9, 7, 8, 11, 12, 2, 14, 4, 1]. These are the motivated distributed management models. A typical management model is called Management by Delegation (MbD) [13, 9, 7, 8, 11], in which management programs describing some of management tasks are executed on managed systems such as routers and switches. Another model is management by mobile agents (MA) [12, 2, 14, 4, 1], in which mobile agents with some management tasks migrate to/from management and managed systems. They provide a means to enhance scalability by distributing some management tasks to the managed systems, and by reducing the management traffic overhead with appropriate filters at the managed systems.

In the distributed network management, dynamic load balancing across an entire managed network is one of the most important challenges in which the management programs should be distributed and executed on management and managed systems, when considering dynamically changing network configurations and resource utilization. In a theoretical sense, the problem of finding an optimal subset of systems on which the management programs should be distributed, such that a given objective function associated with resource utilization is minimized, is known as a ”p-center” or a ”p-median” problem, and both are NP-hard [3]. Some approximation methods do not always provide optimal solutions but offer practical, acceptable solutions, which have recently been studied [6, 5]. Yet, they cannot adequately perform dynamic load balancing. Since the criteria for load balancing is the number of management programs and is not associated with system processing load, the method [6] cannot perform dynamic load balancing in terms of network resource utilization. Although another criterion is associated with the processing load, the method [5] can be performed only within a localized subnetwork and cannot strike a load balance across an entire managed network.

For a solution to this, this paper proposes a new dynamic load balancing method for distributed network management. It allows us to strike a load balance between management and managed systems across an entire managed network, in considering the following two criteria: 1) processing load of the management and managed systems, and 2) bandwidth required for executing all management programs. A typical example of an entire managed network is a set of subnetworks connected together with links like Ethernet and managed by a single organization. The proposed method is realized by attempting to limit the value of a given function in terms of CPU utilization of systems within a small specified value, as well as limiting bandwidth required to execute all management programs within another specified value. In terms of functions, two typical algorithms based on the proposed method are introduced, each having different functions: 1) mean deviation function, and 2) range function. Mean deviation function is defined as the absolute difference between the CPU utilization of each system and the average CPU utilization for all systems, while range function is defined as the difference between the maximum and minimum CPU utilization for all systems.

The algorithms are evaluated analytically according to: 1) setting appropriate limiting values for effective load balancing, 2) capability of two algorithms in performing load balancing and 3) time complexity of two algorithms. Next, a prototype system is implemented based on the proposed method with the stronger algorithm, and evaluated empirically by applying the system to an operational LAN according to: 1) deriving suitable limiting values used in the method, and 2) whether the proposed method can perform well using the derived value, with trivial overhead.

The rest of the paper is organized as follows: Section 2 presents an overview of the distributed network management. It describes two existing methods and addresses their disadvantages with respect to dynamic load balancing in Section 3. In Section 4, a new dynamic load balancing method is proposed, and a prototype system is implemented in Section 5. Two typical algorithms derived from the proposed method are evaluated analytically while evaluating the proposed method in an operational LAN empirically in Section 6.

2. Overview of Distributed Network Management

The distributed management model assumes a managed system such as routers and switches with sufficient computational resources to execute management programs or mobile agents. As a mobile agent may be viewed as a management program capable of traversing one or more managed systems in a specified order, this section describes an overview of how the management program is deployed below.

A management system distributes a management program describing some of management tasks and delegates the task to a managed system as shown in Figure 1(1). Typical management programs sometimes collect management information from MIB (Management Information Base) in the managed system, by polling limited to within the managed system (2). Others aggregate, analyze or filter collected information (3),

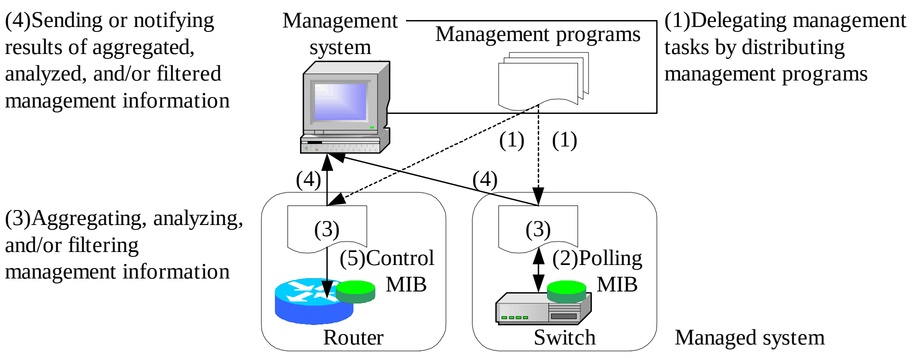

(4)Sending or notifying results of aggregated, analyzed, and/or filtered management information
Management system
Management programs
(1)Delegating management tasks by distributing management programs
(1)
(1)
(4)
(4)
(3)Aggregating, analyzing, and/or filtering management information
(3)
(5)Control MIB
(3)
(2)Polling MIB
Router
Switch
Managed system

Figure 1. Overview of distributed network management

send the information or notification to the management system (4) and control the managed systems according to the results (5). The distributed network management provides a promising means to improve scalability of centralized management models, typically through the distribution of tasks and by reducing management traffic overhead.

3. Existing Methods and Disadvantages

In the distributed network management, dynamic load balancing is a key challenge. For this purpose, some methods have been studied as described below.

3.1 Existing Methods

3.1.1 Method Based on Number of Management Programs. This method [6] performs load balancing such that the number of managed systems assigned to each management program may be almost the same when the number of management programs has been changed by executing a new management program and by terminating existing management programs, or on a regular basis.

When a new management program is executed, this 'leader' management program directs an entire load balancing process. First, the leader collects information required for the process from all management programs, including distances between each management program and the leader, and identifiers of managed systems that each management program currently manages. Next, the leader determines managed systems to be assigned to the leader in order of proximity, until the number of managed systems assigned to each management program may be almost the same. Then, the leader determines managed systems to be assigned to other management programs in the same manner. Finally, the leader makes notification on process termination with information covering all changes.

3.1.2 Method Based on Number of Managed Systems. In the method [5], an entire managed network is divided into some subnetworks such that the number of managed systems in each subnetwork is less than or equal to a specified threshold. A management program is deployed per subnetwork and load balancing is also performed per subnetwork.

When a new managed system is attached to a subnetwork and the number of managed systems in the subnetwork exceeds the threshold, this subnetwork is divided into two subnetworks. A new management program is executed on a managed system in one subnetwork, while the existing management program remains on a managed system in another subnetwork. The two management programs manage each subnetwork independently.

The management program can move to another managed system a within a subnetwork it is responsible for from their current managed system b, if $U_a < (1 - r)$

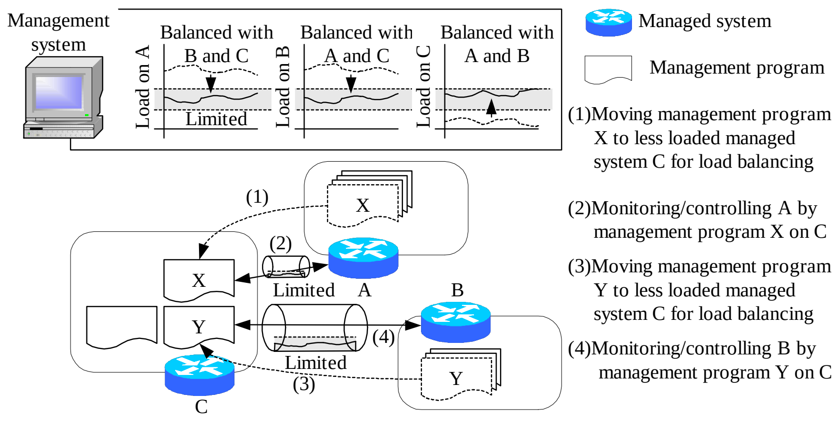

Management system
Balanced with B and C
Balanced with A and C
Balanced with A and B
Load on A
Load on B
Load on C
Limited
Managed system
Management program
(1)Moving management program X to less loaded managed system C for load balancing
(2)Monitoring/controlling A by management program X on C
(3)Moving management program Y to less loaded managed system C for load balancing
(4)Monitoring/controlling B by management program Y on C
(1)
(2)
(3)
(4)
X
Y
Limited
A
B
C

Figure 2. Principle of proposed method

U_b, where U_x denotes a linear function of CPU and memory utilization of managed system x, and $r \in (0, 1)$ is constant. CPU and memory utilization are collected by using another probe management program in the subnetwork.

3.2 Disadvantages

The first method based on the number of managed programs certainly takes note of network configurations such as distance, yet it cannot perform dynamic load balancing sufficiently as load balancing criteria is not at all associated with dynamically changing network resource utilization such as processing load of management and managed systems.

The second method based on the number of managed systems performs dynamic load balancing to a certain extent, in considering both network configuration and resource utilization; however, the method performs only within a localized subnetwork and thus, cannot strike a load balance across an entire managed network. Even if the method could be applied to an entire managed network, the function would be too simple to perform closer load balancing where the absolute differences between CPU utilization of any two systems should be small, as will be analyzed in Section 6.

For a solution to this, a new dynamic load balancing method is proposed for the distributed network management below, which incorporates dynamically changing network configuration and resource utilization, and strikes a load balance between systems across an entire managed network.

4. Proposed Method

4.1 Design Principle

4.1.1 Coexistence of Distributed and Centralized Management Models. As described in Section 2, with respect to processing load associated with management tasks on management and managed systems, as well as management traffic overhead, the distributed management model complements the centralized one where management tasks including management information gathering via polling are concentrated to a management system, while processing load associated with management tasks on managed systems can be minimized. In the proposed method, the distributed and centralized management models coexist, and load balancing is achieved by making full use of both models.

As shown in Figure 2, management programs are executed not only on direct target managed systems to be monitored or controlled, but also on less loaded other systems, including management systems. Suppose the management programs X and Y are responsible for monitoring managed systems A and B, respectively. If the managed system C is loaded less than A, the method moves X from A to C for load balancing (Figure 2(1)). After that, X resumes monitoring their direct target managed system A from C by polling or with management operations via a network similar to management systems in the centralized management model (2). If managed system C is still loaded less than B, the method moves Y from B to C (3). Y resumes monitoring their direct target managed system B from C in the same manner as X (4).

4.1.2 Processing Load of Management and Managed Systems as First Criteria for Load Balancing. The CPU utilization associated with processing load of management and managed systems is used as the first criteria for load balancing. A new threshold of CPU utilization is introduced to detect a system overload. The proposed method achieves load balancing by limiting a value of a given function in terms of CPU utilization within a given threshold as shown in Figure 2. A variety of functions, from simple to complex ones, are possible. Two typical and basic functions will be defined in Section 4.2.1.

4.1.3 Bandwidth Required for Executing all Management Programs in Managed Network as Second Criteria for Load Balancing. Some management programs may be executed on a less loaded, indirect target managed system for monitoring their direct target managed system by polling or management operations through a network in the proposed method. This may cause additional management traffic. The bandwidth required for executing all management programs in a managed network is used as the second criteria to prevent the traffic. A new bandwidth threshold is introduced, and the proposed method is realized by limiting the bandwidth to within the given threshold as shown in Figure 2

4.1.4 Load Balancing by Management System. The management system is responsible for an entire load balancing process in the proposed method in order for load balancing to occur across an entire managed network. The system collects information such as system processing loads, updates information, determines management programs to be moved and destination systems when system overloads are detected, and performs load balancing in accordance with the determination. Leaders may be changed in every process as described in Section 3.1.1, though the process may become more complicated due to information and status synchronization between management programs.

4.2 Dynamic Load Balancing by Proposed Method

4.2.1 Defining Criteria for Load Balancing. The proposed method uses 1) processing load of management and managed systems, and 2) bandwidth required for executing all management programs as criteria for load balancing. The following two basic functions from each type are defined for the first criteria associated with processing load as the variability indices of system CPU utilization can be mainly classified into deviation and range types. For the sake of simplicity, the computational power of all nodes is assumed as the same in the following definitions. Note that it may be better to use stronger nodes over weaker ones. Definitions should be changed in this instance. A typical modification is weighted CPU utilization multiplied by some factor depending on the calculation power.

Mean Deviation Function. Let $c_{i,T}$ (%) be the average system CPU utilization i ($1 \leq i \leq N$) over T seconds, where N denotes the number of management and managed systems. As provided by Equation (1), the mean deviation function $\epsilon_{i,T}$ is defined as the deviation of $c_{i,T}$ from the average (mean) value of all $c_{j,T}$'s ($1 \leq j$

$\leq N$) systems. Note that the average for all CPU utilization or maximum utilization should be the CPU utilization for a multi-CPU system.

$$\epsilon_{i,T} \overset{\text{def}}{=} \left| c_{i,T} - \left(\Sigma_{j=1}^{N} c_{j,T} \right) / N \right| \quad (\%). \tag{1}$$

Range Function. As given in Equation (2), the range function is defined as the difference between the maximum and minimum CPU utilization for all systems.

$$\delta_T \overset{\text{def}}{=} \max_i c_{i,T} - \min_i c_{i,T} \quad (\%). \tag{2}$$

If the proposed method cannot make determination with only two criteria, priority is assigned to each management program by a network operator.

4.2.2 Determining Management Program and Destination System.

Two thresholds are introduced here. One is tolerable variation Δ $(\%)$ and another is tolerable bandwidth B (bps). The proposed method attempts to limit all $\epsilon_{i,T}$s' or δ_T within Δ, as well as bandwidth required for executing all management programs within B. They are specified by a network operator. How to set appropriate values of these thresholds for effective load balancing will be evaluated in Section 6.1.1. For the sake of simplicity, the rest of this section focuses attention on $\epsilon_{i,T}$, and the same holds for δ_T.

As candidates for load balancing, the proposed method first selects management programs on the most loaded system i where $\epsilon_{i,T} > \Delta$ and $c_{i,T}$ is the maximum for all the systems. Next, the proposed method determines the system j with a minimum $c_{j,T}$ as the destination system, as well as the management program to be moved from the candidates for load balancing, such that the decrease in bandwidth required for executing all management programs after moving towards the destination system is maximized. If there is no such management program, it determines one where the increase is minimized.

The management system notifies the network operator and terminates the execution of the management program with the lowest priority on the most loaded system if any management program movement results in excess bandwidth over the tolerable bandwidth B.

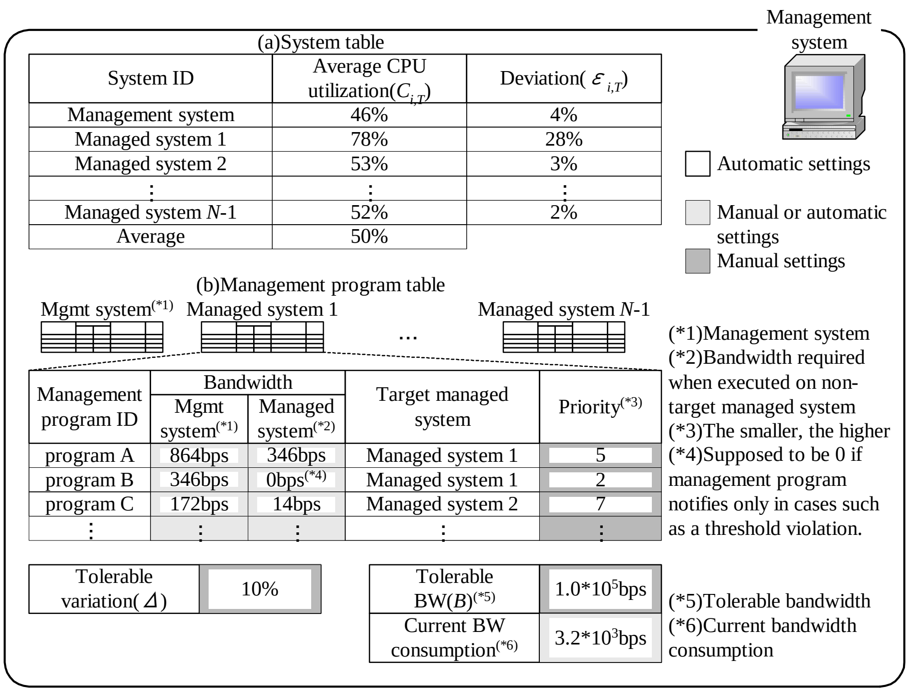
4.2.3 Performing Load Balancing.

The management system moves and resumes the management program to the destination system in accordance with the determination as described in Section 4.2.2. The management system notifies a network operator of a failure if move or resume fails for some reason. The management system updates information required for a subsequent load balancing process such as System table, Management program tables and current bandwidth consumption as described in Section 4.2.4.

4.2.4 Managing Information for Load Balancing.

Figure 3 shows all information required for load balancing managed by the management system, including (a) System table, (b) Management program tables, tolerable variation Δ, tolerable bandwidth B and current bandwidth consumption.

System table maintains average CPU utilization for each system and derives mean deviation $\epsilon_{i,T}$ from utilization. The management system distributes the CPU monitoring program to each system to monitor CPU utilization. The management system sets $c_{i,T}$ to System table obtained by CPU monitoring programs. The Management program table is provided for each system to maintain information on management programs executed on the system. It includes the identifier of a management program executed on the system, the bandwidth required when executed on the management


Management system
Mgmt system(*1)
Managed system 1
…
Managed system N-1

(a)System table

System ID	Average CPU utilization($C_{i,T}$)	Deviation($\varepsilon_{i,T}$)
Management system	46%	4%
Managed system 1	78%	28%
Managed system 2	53%	3%
⋮	⋮	⋮
Managed system N-1	52%	2%
Average	50%	

□	Automatic settings
▨	Manual or automatic settings
■	Manual settings

(b)Management program table

Management program ID	Bandwidth		Target managed system	Priority(*3)
	Mgmt system(*1)	Managed system(*2)		
program A	864bps	346bps	Managed system 1	5
program B	346bps	0bps(*4)	Managed system 1	2
program C	172bps	14bps	Managed system 2	7
⋮	⋮	⋮	⋮	⋮

Tolerable variation(Δ)	10%

Tolerable BW(B)(*5)	$1.0*10^5$bps
Current BW consumption(*6)	$3.2*10^3$bps

(*1)Management system
(*2)Bandwidth required when executed on non-target managed system
(*3)The smaller, the higher
(*4)Supposed to be 0 if management program notifies only in cases such as a threshold violation.

(*5)Tolerable bandwidth
(*6)Current bandwidth consumption

Figure 3. Operational information and settings required for load balancing

system and on managed systems, the direct target managed system for a given management task and priority.

The current bandwidth consumption is summed up by the bandwidth consumed by all management programs executed, which is maintained by Management program tables. The bandwidth consumed by each management program is manually or automatically derived. For example, if a management program uses a specific management protocol such as SNMP, the bandwidth can be roughly derived from the polling interval and the expected length of PDU (Protocol Data Unit), including the number of pieces of management information, their syntax and expected returned values.

4.3 Example

We show how the proposed method performs load balance below, with operational information and settings in Figure 3 and the flowchart in Figure 4.

The tolerable variation Δ and tolerable bandwidth B are set 10% and $1.0 * 10^5$ bps, respectively (Figure 4 S1). The CPU monitoring program is executed on each system (S2). Then, the method configures System table, as well as Management program tables (S3), and distributes management programs on direct target systems and executes them (S4). The method attempts to perform load balancing, since $\epsilon_{1,T}$ $(= 28\%) > \Delta$ $(= 10\%)$ (S5). The method selects management programs on the most loaded system 1 as the load balancing target system (S6). The method determines the management system with the minimum average CPU utilization as a destination system (S7) and management program C as the load balancing target, resulting in a maximum 14 bps (from 172 + 14 bps to 172 bps) decrease in bandwidth consumption (S8). Since the resulting bandwidth consumption does not yet exceed the tolerable bandwidth B (S9), the method performs load balancing by oving the management program C from sys-

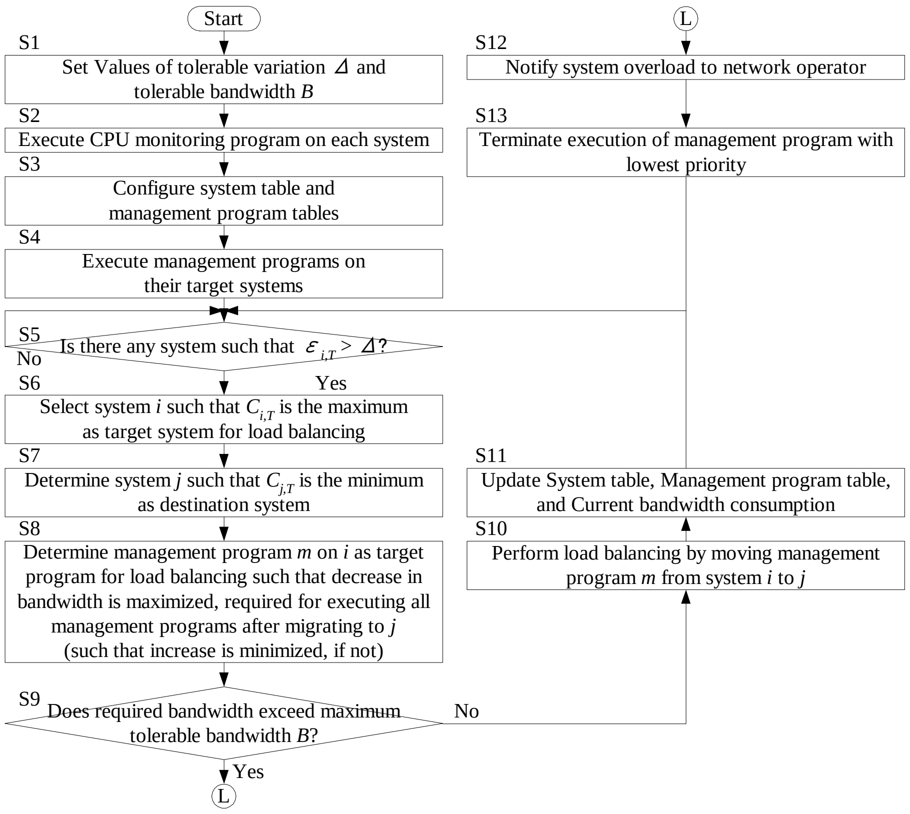

Start
S1
Set Values of tolerable variation Δ and tolerable bandwidth B
S2
Execute CPU monitoring program on each system
S3
Configure system table and management program tables
S4
Execute management programs on their target systems
S5
Is there any system such that ε i,T > Δ?
No
Yes
S6
Select system i such that C i,T is the maximum as target system for load balancing
S7
Determine system j such that C j,T is the minimum as destination system
S8
Determine management program m on i as target program for load balancing such that decrease in bandwidth is maximized, required for executing all management programs after migrating to j (such that increase is minimized, if not)
S9
Does required bandwidth exceed maximum tolerable bandwidth B?
No
Yes
L
S12
Notify system overload to network operator
S13
Terminate execution of management program with lowest priority
S11
Update System table, Management program table, and Current bandwidth consumption
S10
Perform load balancing by moving management program m from system i to j

Figure 4. Flowchart of proposed method

tem 1 to the management system (S10). Finally, the method returns the process to S5 after updating the tables and current bandwidth consumption (S11).

If the resulting bandwidth consumption exceeds the tolerable bandwidth B (S9), the method does not move management program C while notifying the network operator of system overloads (S12). The method terminates execution of the management program with the lowest priority on the system (S13), and returns the process to S5.

5. Prototyping

A prototype system is implemented in Java. Currently, minimum functions sufficient for evaluating the proposed method are implemented. Figure 5 shows the system diagram. The system runs on a PC, which emulates a managed system with sufficient computational resources for a management program. We use legacy SNMP agent, and AdventNet, Inc. SNMPv3 Package2.2 for Java API to the SNMP agent. We give MIB definition for initial settings. There can be many varieties of management programs with different management tasks, and two typical management programs [11] are implemented, in addition to the CPU monitoring program as in Table 1.

These management programs can be generated easily by specifying a few parameters (Figure 5 (1)). The management application then approximates the bandwidth required for executing each management program and registers them with Management program tables in Figure 3 (b). The download and unload, start, suspend and resume of management program execution are realized by Java RMI (Remote Method Invocation). When execution starts (3), polling to the legacy SNMP agent is performed

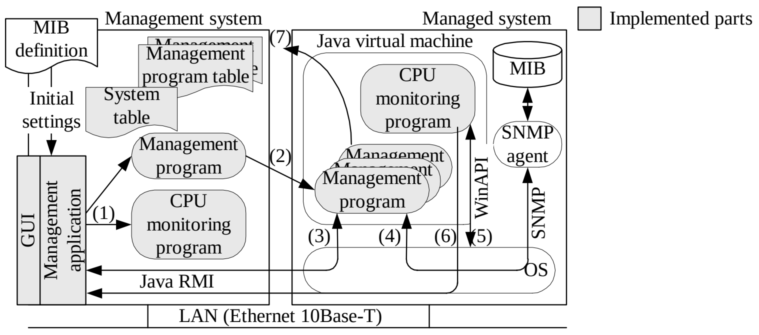


Figure 5. Prototype system diagram

Table 1. Management programs in prototyping.

Management program	Description	Parameters (not exhaustive)
A	notifies threshold violation to management system on result of polling	direct target managed system, names of management information for polling[a], polling interval[b], upper and lower thresholds[b]
B	sends aggregated management information collected by polling for every specified time to management system	direct target managed system, names of management information for collection[a], polling interval[b], sending interval[b]
CPU monitoring program	sends average CPU utilization collected by polling for every specified time to management system	direct target managed system, polling interval, sending interval

[a]Possible to specify multiple names.

[b]Possible to specify a different value for each piece of management information.

according to the specified parameters (4). Threshold violations and aggregated management information are also notified and sent by RMI (3). As CPU utilization is not defined in a standard MIB, it is collected through OS-dependent API (5) and sent by RMI (6). If heavy processing load on a system is detected, management program execution is suspended and the management program is moved to another less loaded system with its execution context (7).

6. Evaluations

6.1 Analytical Evaluations

This section evaluates two algorithms A_d and A_r, each obtained by applying Eq.(1) and Eq.(2) to the proposed method according to: 1) appropriate tolerable variation value Δ and tolerable bandwidth B for effective load balancing in Section 6.1.1, 2) capability of algorithms on their ability to perform load balancing where the sum of the absolute differences between CPU utilization of any two systems should be small in Section 6.1.2, and 3) time complexity of Eq.(1) and Eq.(2) in Section 6.1.3.

6.1.1 Tolerable Variation Δ and Bandwidth B.

Tolerable Variation Δ. The smaller the value of tolerable variation Δ, the smaller the absolute difference between CPU utilization of any two systems $c_{i,T}$ and $c_{j,T}$ ($1 \leq i, j \leq N$, $i \neq j$), thus the algorithms can more closely strike a load balance among systems across an entire managed network. However, if the value of Δ is set too small, even a slight difference between two systems may trigger a load balancing process causing undesirable effects where the algorithms continue to move a management program from system to system. The optimal value of Δ preventing the effect should be the minimum one satisfying the following two conditions simultaneously.

Condition1	The value of Δ is greater than the maximum processing load of all management programs for all systems.
Condition2	The value of Δ is greater than the maximum processing load of all destination systems when a management program moves there.

Assuming a case where only a single management program is executed on a system across an entire network, and the average CPU utilization of all other systems is 0%, unless Condition 1 holds, the execution of the management program on any system triggers a load balancing process and causes the effect. It can be proven that Δ satisfying Condition 1 for the above case can prevent the effect in any other case. For algorithms A_d and A_r, such Δ is provided by Equation (3), where N and $c_{\max}$ denote the number of systems and maximum processing load of all management programs for all systems, respectively.

$$\Delta > \begin{cases} c_{\max}\,(1 - 1/N) & \text{for } A_d, \\ c_{\max} & \text{for } A_r. \end{cases} \tag{3}$$

Unless Condition2 holds, the increasing processing load of the destination system in moving a management program for the load balancing triggers another load balancing process and this also causes the effect. This can be avoided by a time window, for which the method is prohibited from a resulting load balancing process.

Tolerable Bandwidth B. Since user traffic and management traffic share bandwidth in LAN and WAN networks, it is generally desirable that the bandwidth for management traffic be restricted at most to 5% of the minimum bandwidth of the network [10]. Accordingly, for example, B is set to 100 Kbps, which is 5% of the effective bandwidth 2 Mbps of a 10Mbps Ethernet LAN.

6.1.2 Close Load Balancing Capability.

Here, an evaluation function defined by Eq.(4) is introduced to show how closely the algorithms A_d and A_r can perform load balancing. The smaller the value of the evaluation function is, the closer load balancing an algorithm can perform, where the absolute differences between CPU utilization of any two systems is small.

$$\Sigma_{1\le i<j\le N} \left| c_{i,T} - c_{j,T} \right| . \tag{4}$$

If the algorithms do not perform the load balancing process for a sufficient amount of time, Eq.(5) holds. With Eq.(5), the upper limits of the evaluation function Eq.(4) of the algorithms A_d and A_r are provided by Eq.(6) and Figure 6 for illustrative purposes.

$$\begin{aligned} &\left| c_{i,T} - \left(\Sigma_{j=1}^{N} c_{j,T}\right)/N \right| \le \Delta, && \text{for } A_d, \\ &\max_i c_{i,T} - \min_i c_{i,T} \le \Delta, && \text{for } A_r. \end{aligned} \tag{5}$$

Since equality of Eq.(6) holds for the algorithm A_r where half of the $c_{i,T} = \Delta$ and the rest of $c_{i,T} = 0$, and the upper limit of the algorithm A_r is tight, the algorithm A_d can perform load balancing more closely than the algorithm A_r.

The result shows that the mean deviation function is more suitable than the range function for the dynamic load balancing method striking a load balance across an entire managed network. Recall that the linear function defined in the existing method [5] is a variation of Eq.(2). This implies that even if the existing method [5] could be applied to an entire managed network, there would still be room for improvement, and it would be better to use the mean deviation function for closer load balancing.

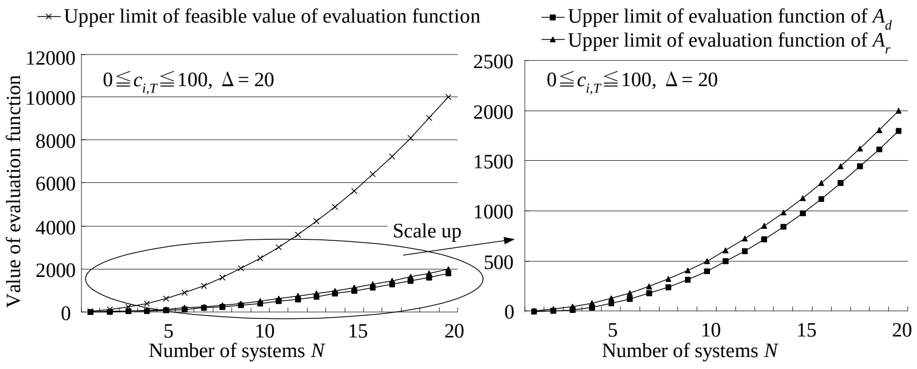


Figure 6. Capability of load balancing of algorithms

$$\Sigma_{1 \le i < j \le N} |c_{i,T} - c_{j,T}| \le \begin{cases} 1/4N(N-2)\Delta & \text{for } A_d, \\ 1/4N^2\Delta & \text{for } A_r. \end{cases} \qquad (6)$$

6.1.3 Time Complexity of Criteria for Load Balancing.

Time Complexity of Mean Deviation Function. An evaluation of mean deviation function Eq.(1) requires $(2N + 1)$ additions and one division. The time complexity of the mean deviation function is $O(N \log |c_{i,T}| + (\log |c_{i,T}|)^2)$ as the time complexity of an addition and division are $O(\log |c_{i,T}|)$ and $O((\log |c_{i,T}|)^2)$, respectively.

Time Complexity of Range Function. An evaluation of mean deviation function Eq.(2) requires one max, one min and one additional operation. The time complexity of the range function is $O(N + \log |c_{i,T}|)$ as the time complexity of max and min operations over N elements is both $O(N)$.

The above results show that time complexity of the range function is better than that of the mean deviation function. However, since $|c_{i,T}|$ represents system i processing load and is limited by at most $100(\%)$ in reality, the number of N systems dominates both complexities; thus, they can be considered the same as $O(N)$. Along with the result in 6.1.2, the algorithm A_d with the mean deviation function performs better than A_r at almost the same computational cost.

6.2 Empirical Evaluations

By applying the prototype system to an operational LAN, how well the proposed method can perform load balancing with a trivial overhead is evaluated. Based on the results in Section 6.1, the mean deviation function is used in this section. The system specification in the evaluation is shown in Table 2. The parameter values of management programs are set as in Table 3. All PCs are connected to the same LAN (10Mbps Ethernet) and the maximum tolerable bandwidth B is set to 100Kbps. The proposed method attempts to perform load balancing every 100 seconds.

6.2.1 Tolerable Deviation Δ in Operational LAN. First, the minimum value is derived of the tolerable deviation Δ satisfying Condition1 in Section 6.1.1. The processing load is 65.0% when management program A polling 20 managed ob-

Table 2. System specification[a] in evaluation.

Name	CPU	CPU frequency (MHz)	Memory (MB)	Description
System1	Intel Pentium	133	32	managed system
System2	Intel Pentium	150	96	managed system
System3	Intel Pentium II	266	64	managed system
System4	Intel Pentium II	450	64	managed system
System5	Intel Pentium II	450	128	management system

[a]OS is WindowsNT4.0 SP5.

Table 3. Parameter settings of management program A and CPU monitoring program.

Management program A	
Management information for polling	Object types of counter syntax in MIB-II ifEntry
Number of managed object instances for polling	1, 5, 10, 15 and 20
Polling interval[a]	5, 10, 15, 30, 60 and 180 seconds
Upper and lower thresholds	500 and 0
Priority	Common to all management programs
CPU monitoring program	
Polling interval	1 second
Sending interval	100 seconds

[a]Common polling interval when more than one object instance is specified.

ject instances for every 5 seconds and referred to as $A_{20,5}$ is executed on System1 with the lowest processing capability of all systems. This implies that $c_{\max}$=65.0%. By Eq.(3), $\Delta>65.0\times(1-1/5)=52.0(\%)$. This value is obviously too greater. A smaller value can be obtained by more specific and finer-grained management programs. That is, 20 management programs each polling one managed object instance every 5 seconds and referred to as $A_{1,5}$, perform an almost equivalent management task together with that of an $A_{20,5}$. It brings a smaller value of Δ for closer load balancing. In this evaluation, there is hardly any difference betaween $A_{20,5}$ and 20 $A_{1,5}$'s processing loads for any other values for the number of managed object instances for the polling and polling interval. By executing the more specific and finer-grained management programs, i.e., 20 $A_{1,5}$'s on System1, $c_{\max}$ in turn becomes 4.2%, then the value of Δ is reduced to $4.2\times(1-1/5)=3.36\%$.

Next, the minimum value of Δ satisfying Condition2 is derived. It takes 18.7, 21.5, 6.5 and 4.1 seconds on average to download the management program $A_{20,5}$ to System1, 2, 3 and 4. This time is almost the same for any other values of the number of managed object instances for the polling and polling interval. For every download, the CPU utilization marks 100% all the time and this implies that these download times correspond to the processing load per 100 seconds. The maximum processing load of all destination systems where a management program is downloaded is 21.5% and $\Delta > 21.5\%$ when T is 100 seconds.

From the above discussion, the minimum value of Δ satisfying both conditions simultaneously is 21.5%.

6.2.2 Load Balancing by Proposed Method in Operational LAN.

A heavy load is imposed on System2 by executing 20 management programs, each polling one managed object instance every 5 seconds, as well as 25% stationary processing load. Measurement is performed for the first time when there is no management program moved by the proposed method for the last 10 minutes, referred to as convergence time, and the maximum deviation of the average CPU utilization at convergence time, for some different values of Δ. The method attempts to move a management program every 100 seconds in sequence and all the 20 management pro-

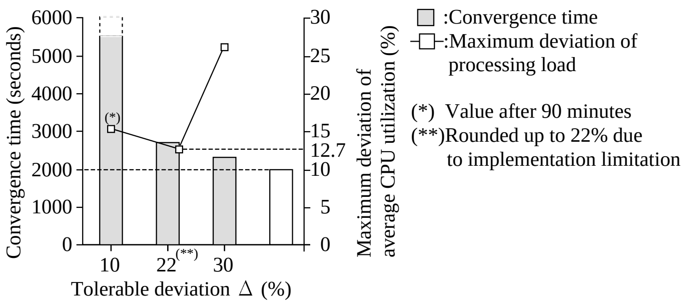


Figure 7. Convergence time and maximum deviation of average CPU utilization

grams being distributed to other systems is optimally desirable. Thus, it takes at least 2000 seconds ($=20\times100$) for any value of Δ, shown as the white bar for comparison in Figure 7.

As shown in Figure 7, the convergence time shortens as the value of Δ is greater, whereas the maximum deviation of average CPU utilization increases from value 22% to 30%. The convergence time could not be measured within 90 minutes ($=5400$ seconds) in the worst case when the value of Δ is 10%, smaller than the value of 22%. All of the above results show that setting the value of $\Delta = 22\%$ can control the differences in average CPU utilization between systems within a small value of 12.7% as in Figure 7. Thus, the proposed method can strike a closer load balance for management tasks.

6.2.3 Bandwidth Consumption. The CPU monitoring program has small bandwidth consumption for sending CPU utilization messages. It is a one-way message at 60 bytes. It is smaller than the size of a minimum SNMP PDU of approximately 90 bytes, and becomes 180 bytes with a request and response pair. In this evaluation, since the sending interval is 100 seconds, the bandwidth consumption is 4.8bps. When the number of managed systems is 1000, then it becomes and is sufficiently small enough if compared with the tolerable 4.8Kbps bandwidth B ($=$ 100Kbps) in Section $6.1.1$.

6.2.4 Processing Load of Management and Managed Systems. The load balancing overhead of the proposed method on the management system is up to 16.2% and the process is completed within 1.4 seconds. This is approximately equivalent to that of a single polling for 12 managed object instances from the management system in the evaluation. Since the management system performs polling in the tens of managed object instances in general, the proposed method does not adversely affect the advantage of the distributed management model.

The CPU monitoring program processing load is 2.3%, 2.1%, 0.9% and 0.5%, respectively, when executed on System1, 2, 3 and 4. Each of them is less than or equal to the processing load of the management program A, polling a managed object instance every 5 seconds when executed on the corresponding systems. This brings hardly any impacts on the managed system in practical.

7. Conclusions

This paper proposed a new dynamic load balancing method for distributed network management and evaluated it both analytically and empirically. The proposed method strikes the load balance of management tasks across an entire managed network on the basis of 1) the processing load of management and managed systems, and 2) the bandwidth required for executing all management programs through the coexistence of centralized and distributed management models.

Since the criteria for the load balancing of an existing method is not at all associated with the processing load of management and managed systems, it cannot perform dynamic load balancing in considering network resource utilization. Although there is another existing method whose criteria are associated with the processing load, the method can perform only within a localized subnetwork and cannot strike a load balance across an entire managed network. Moreover, the existing method is less capable, as some load balancing functions are too simple to perform closer load balancing. The upper limit of absolute differences between CPU utilization of any two systems of the existing method is at most $N^2/4$, where N denotes the number of systems across an entire managed network. By introducing the mean deviation function as a new load balancing criteria, the proposed method can improve the capability and result in absolute differences between CPU utilization of any two systems at up to $N(N-2)/4$, with the same calculational cost $O(N)$.

A prototype system was also implemented based on the proposed method and evaluated through application in an operational LAN. The results showed that the proposed method could perform well in practice with only a slight overhead. An extension to wide area network and more evaluations in heterogeneous networks are left for further studies.

Acknowledgments

We would like to thank Mr. Tohru Asami, President, CEO of KDDI R&D Laboratories Inc., Dr. Yuichi Mastushima, Executive Vice President and Dr. Takeshi Mizuike Executive Director for their continuous encouragement with respect to this research.

References

[1] P. Bellavista, A. Corradi, and C. Stefanelli. An Open Secure Mobile Agent Framework for Systems Management. *Network System Management*, Vol.7, No.3, pp.323–339.

[2] A. Bieszczad, B. Pagurek, and T. White. Mobile Agents for Network Management. *IEEE Comm. Surveys*, Vol.1, No.1, 1998.

[3] M. Garey and D. Johnson. *Computers and Intractability, A guide to the Theory of NP-completeness*. W.H. Freeman and Co., 1979.

[4] D. Gavalas, D. Greenwood, M. Ghanbari, and M. O'Mahony. An Infrastructure for Distributed and Dynamic Network Management based on Mobile Agent Technology. In *Proc. of IEEE ICC '99*, pp.1362–1366, 1999.

[5] D. Gavalas, D. Greenwood, M. Ghanbari, and M. O'Mahony. Hierarchical Network Management: A Scalable and Dynamic Mobile Agent-based Approach. *Computer Networks*, Vol.38, No.6, pp.693–711, 2002.

[6] R. Giladi and M. Gat. Meta-management of Dynamic Distributed Network Managers. In *Proc. of IFIP/IEEE DSOM* 2000, pp.119–131, 2000.

[7] G. Goldszmidt and Y. Yemini. Evaluating Management Decisions via Delegation. In *Proc. of IFIP ISINM* ′93, pp.247–257, 1993.

[8] G. Goldszmidt and Y. Yemini. Delegated Agents for Network Management. *IEEE Comm. Mag.*, Vol.36, No.3, pp.66–70, 1998.

[9] J. Grégoire. Models and Support Mechanisms for Distributed Management. In *Proc. of IFIP ISINM* ′95, pp.17–28, 1995.

[10] IETF RFC 1157. *A Simple Network Management Protocol (SNMP)*, May 1990.

[11] A. Liotta, G. Knight, and G. Pavlou. Modelling Network and System Monitoring over The Internet with Mobile Agents. In *Proc. of IFIP/IEEE NOMS* ′98, pp.303–312, 1998.

[12] T. Magedanz and T. Eckardt. Mobile Software Agents: A New Paradigm for Telecommunication Management. In *Proc. of IFIP/IEEE NOMS* ′96, pp.360–369, 1996.

[13] Y. Yemini, G. Goldszmidt, and S. Yemini. Network Management by Delegation. In *Proc. of IFIP ISINM* ′91, pp.95–107, 1991.

[14] M. Zapf, K. Herrmann, and K. Geihs. Decentralized SNMP Management with Mobile Agents. In *Proc. of IFIP/IEEE IM* ′99, pp.623–635, 1999.