Reprinted from IM2001

Models and Analysis of Trade-offs in Distributed Network Management Approaches

Y. Zhu
Microsoft Corp.
One Microsoft Way
Redmond, WA
USA
yuhangz@microsoft.com

T. Chen
SMU, Dept of EE
PO Box 750338
Dallas, TX
USA
tchen@seas.smu.edu

S. Liu
Verizon Labs, Inc.
40 Sylvan Road, MS-48
Waltham, MA
USA
steve.liu@verizon.com

Abstract

This paper develops an analytical framework to model and compare different distributed approaches for network monitoring. The performance of each scheme is evaluated in terms of traffic, completion time, and processing and memory resources. It is shown that distributed approaches have considerable advantages over traditional centralized network management, but a single distributed approach may not be best for all applications. Analytical results suggest that the weak mobility approach performs well for routine monitoring, and the mobile agent approach is well suited for searching types of applications. However, the best approach for a given application will depend on many parameters requiring careful quantification.

Keywords

Distributed network management, mobile agents

1. Introduction

Network management is mainly concerned with the practical problem of monitoring networks and maintaining their integrity and performance. Network management systems today are predominantly based on the SNMP (simple network management protocol) paradigm [1]. In the SNMP paradigm, a centralized network management system (NMS) communicates with simple agents residing in each managed node. This approach is clearly inefficient for certain applications. For example, the potential processing and traffic bottleneck at the NMS can prevent scalability to large networks. If the NMS is searching for data, it will poll nodes repeatedly until the data is found, involving many messages through the network and processing burden concentrated at the NMS.

On the other hand, the processing capabilities of most network nodes have vastly improved in the past few years, making possible sophisticated agents with intelligence or mobility. There are major potential advantages to distributing and automating

0-7803-6719-7/01/$10.00 (c) 2001 IEEE

network management functions. First, data processing and searching can be moved closer to the locations where data resides, thereby reducing the network traffic. Second, distributed data processing will eliminate the potential processing bottleneck at the central NMS. Third, automated and distributed data processing may find specific data faster by eliminating the delays involved in repeated polling. Fourth, distributed network management is inherently more robust because it does not rely completely on the central NMS and continuous communications through the network.

Various distributed approaches to network management have been proposed [2-6]. There are trade-offs between different approaches but quantitative comparisons have been very limited. This paper seeks to develop analytical models which can lead to quantitative comparisons of prototypical approaches. The objective of the analytical framework is to help evaluate the most appropriate management scheme for a given management application, e.g., monitoring, searching. Section 2 overviews example distributed approaches and presents the basic models and assumptions for analysis. Section 3 gives an analysis of each management approach for routine monitoring in terms of performance and costs. Section 4 evaluates the trade-offs between the different approaches for a searching application.

2. Distributed Network Management Approaches

Network management functions may be distributed to a hierarchy of mid-level managers, each responsible for managing a usually separate portion of the entire network [2]. The hierarchy may conceptually consist of multiple layers where the lowest level managers still operate by polling managed nodes within its subnetwork and communicate up the hierarchy layers with higher managers. The main advantage compared to non-hierarchical polling is that subnetworks can be managed in parallel, reducing the traffic and processing burden on the highest level NMS. However, the distribution of network management functions and the organization of managers are static. The static structure can be a liability when new management functions need to be initiated or when some management functions are used only infrequently.

In the management by delegation (MBD) approach, management functions may be distributed dynamically by dispatching 'delegated agents' to a managed node where the agent code is executed [3]. This dynamic approach offers the advantage of instantiating management functions only when they are needed, at a cost of additional complexity. Here we assume the mobility of the delegated agent is limited to its transfer from a manager to a managed node, i.e., a 'weak mobility' capability [7].

A 'strong mobility' capability allows an agent to autonomously transfer both code and execution state, usually referred to as a mobile agent [4]. Strong mobility might be expected to be particularly useful for tasks involving searching for data at an unknown location or collection of data that is spread out geographically. The flexibility of mobile agents is achieved at a cost of additional complexity (a common infrastructure to support migration) and more security issues.

For tractability, some simplifying assumptions must be made for the different schemes. Generally, most assumptions are similar to other previous studies of traffic involved in agent-based network management [8-10]. Our study differs from other

work in some details of the models; inclusion of delays and computing resources in the analysis as well as traffic requirements; and the consideration of different types of applications.

- *CS (client-server) model*: As shown in Figure 1(a), the traditional SNMP client-server approach is modeled as a centralized NMS communicating with N network nodes. The communications between the NMS and agents is characterized by pairs of query-response messages for every.
- *HS (hierarchical static) model*: A hierarchical static approach is modeled as L mid-level managers, each managing a separate subnetwork of N/L nodes, as shown in Figure 1(b). A two-level hierarchy is considered here although multiple hierarchical layers may be possible in practice. Each subnetwork is managed by a client-server scheme, and mid-level managers may communicate with a centralized high-level NMS as needed.
- *WM (weak mobility) model*: The NMS distributes code to specific nodes where the code is executed and expires, as shown in Figure 1(c). This approach is natural when management tasks are dynamic and the location of target data is known by the NMS.
- *SM (strong mobility) model*: The NMS dispatches one or more agents to carry out a specific task, as shown in Figure 1(d). The agents have the capability to autonomously travel among different nodes to complete their tasks. Strong mobility includes migration of data (execution state) as well as code. This approach is well suited for applications where the agents must correlate data from multiple nodes or search for an unknown location.

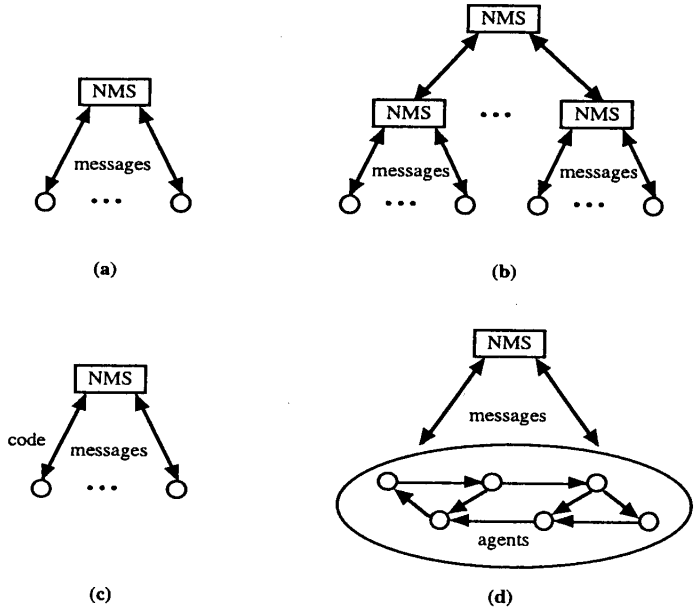

Figure 1: Centralized and distributed network management approaches.

We may expect that no single approach will be best for all possible types of network management applications. Hence the various approaches should be compared in the contexts of different applications, such as: (i) routine network monitoring to detect changes in the network status; (ii) searching for data at a known node location; (iii) searching for data at an unknown node location. We are interested in the performance of each approach in terms of time to complete a specific task and the cost in terms of traffic and computing resources. After developing a model for each approach, it is possible to evaluate the trade-offs among them for network monitoring. Due to the necessary simplifications, there are certain factors (e.g., flexibility, dynamicism, reliability) that are not considered in the analysis here but can be important advantages in actual situations.

3. Routine Network Monitoring

We consider the problem of detecting a random change in a variable X at an arbitrary node. It is assumed that changes in the variable occur randomly as a Poisson process with rate λ at any node within the network, independent of other nodes. Generally, the performance metric of interest is the mean time T to detect the change. The costs include:

- C: amount of traffic (bits/s) measured in total, per node, and per NMS
- M: average amount of memory consumed at nodes
- U: average utilization of processing resources (fraction of total processing time) at nodes and NMS.

3.1 CS Model

It is assumed that the NMS will regularly poll each node to retrieve the value of variable X. Although SNMPv2 allows retrieval of multiple variables in a single query, only a single variable is considered here. The CS approach is characterized by these variables:

- Γ: the polling rate per node ($\Gamma > \lambda$ is assumed)
- I_q: size of each query message (bits)
- I_r: size of each response message (bits)
- t_p: packet delay for query or response message (including transmission and propagation, assumed to be equal between any points in the network)
- t_c: processing time for a message at a node
- t_q: processing time for a query message at NMS
- t_r: processing time for a response message at NMS

The average time until the next query after a variable change is $1/2\Gamma$. The expected time for the NMS to detect a variable change is the sum:

$$T_{CS} = \frac{1}{2\Gamma} + t_c + t_p + t_r \qquad (1)$$

Each managed node will handle $\Gamma(I_q+I_r)$ bits/s. The total traffic in the network is:

$$C_{CS} = N\Gamma(I_q + I_r) \text{ bits/s} \tag{2}$$

The centralized NMS must handle the same total amount of traffic.

In this approach, agents reside permanently at each node. The memory consumption in each node is simply a fixed amount $M_{CS} = B$ bits.

Agents are assumed to consume negligible execution resources except for processing query or response messages. Each node spends a time t_c to process a query or response message in every polling interval $1/\Gamma$. The fraction of total computing time spent by each node is:

$$U_{CS} = 2\Gamma t_c \tag{3}$$

Naturally, this centralized approach imposes heavy computational burden on the NMS. The fraction of total computing time spent by the NMS is:

$$U_{CS} = N\Gamma(t_q + t_r) \tag{4}$$

3.2 HS Model

In the HS model, the previous assumptions are unchanged except some traffic is offloaded from the high-level NMS to L mid-level managers. Within each subnetwork managed by a mid-level manager, the performance is similar to the CS model. With the same polling rate per node, the expected time for a mid-level manager to detect a variable change is the same as (1). If we assume that it takes an additional packet forwarding time t_p and packet processing time t_r to report the variable change to the high-level NMS, then the total expected time for the variable change to be detected by the high-level NMS is:

$$T_{HS} = \frac{1}{2\Gamma} + t_c + 2t_r + 2t_p \tag{5}$$

which is slightly larger than (1).

The manager hierarchy does not effect the operation of each node, so each managed node will handle $\Gamma(I_q + I_r)$ bits/s as before. Each mid-level manager must handle $N\Gamma(I_q + I_r)/L$ bits/s between the N/L nodes that it manages. The mid-level managers are advantageous if they perform some data filtering and report only variable changes to the high-level NMS. The total traffic in the network is:

$$C_{HS} = N\Gamma(I_q + I_r) + N\lambda I \text{ bits/s} \tag{6}$$

Although the total traffic in the network is increased, the hierarchy should reduce the amount of traffic handled by the high-level NMS to $N\lambda I_r$ bits/s. This is certainly a reduction from (2) if $\Gamma > \lambda$ as assumed.

The hierarachy does not change the memory consumption $M_{HS} = B$ bits or utilization of processing resources at each node which is (3). The fraction of total computing time spent by each mid-level manager is:

$$U_{HS} = N\left[\Gamma(t_q + t_r) + \lambda t_r\right] / L \tag{7}$$

The fraction of total computing time spent by the NMS is:

$$U_{HS} = N\lambda t_r \tag{8}$$

which is less than (4).

3.3 WM Model

In the weak mobility approach, for routine monitoring, the code is assumed to reside permanently at the nodes after dispatching. This implies that the one-time cost of dispatching the code will become negligible over the long term. If the agents report only variable changes, the total traffic in the network will be:

$$C_{WM} = N\lambda I_r \text{ bits/s} \tag{9}$$

The NMS handles the same total amount of traffic. The overhead traffic is only the initial code dispatched to nodes; if the code is I_{WM} bits, the overhead is a fixed one-time cost of NI_{WM} bits.

Code resides continuously at each node, so the memory usage per node is a fixed amount $M_{WM} = I_{WM}$.

After the code is dispatched and executed, we assume that the process will sleep and sample the variable at fixed rate Γ. The mean time for the NMS to detect a variable change will be the same as (1). If the process requires t_s processing time for each sample, the utilization of processing resources at nodes will be:

$$U_{WM} = \Gamma t_s \tag{10}$$

The NMS spends a fraction of total processing time equal to

$$U_{WM} = N\lambda t_r \tag{11}$$

3.4 SM Model

For routine monitoring, we assume that L mobile agents travel around the nodes in a preset pattern. Each agent dwells for a time t_s at each node (long enough to sample variable X) and moves to the next node with forwarding delay t_p. If a variable change

is detected, the agent will immediately send a response message to the NMS. The average time to detect a variable change and report it to the NMS is:

$$T_{MA} = \frac{N(t_s + t_p)}{2L} + t_c + t_p + t_r \tag{12}$$

Although mobile agents may generally transfer state as well as code between nodes, this capability is not needed for the routine monitoring application under consideration. Agents are assumed to be a fixed size I_{MA}. The traffic per node is

$$C_{MA} = \frac{2LI_{MA}}{N(t_s + t_p)} + \lambda I_r \text{ bits/s} \tag{13}$$

The total traffic in the network is:

$$C_{MA} = \frac{LI_{MA}}{t_s + t_p} + N\lambda I_r \text{ bits/s} \tag{14}$$

The NMS handles only $N\lambda I_r$ bits/s. Mobility of the agents introduces the additional overhead traffic, at a savings in memory usage per node.

The average memory usage per node is:

$$M_{MA} = \frac{LI_{MA}t_s}{N(t_s + t_p)} \text{ bits} \tag{15}$$

Finally, each node spends a processing time t_s within each interval of $N(t_s + t_p)/L$, implying an average utilization of processing resources:

$$U_{MA} = \frac{Lt_s}{N(t_s + t_p)} \tag{16}$$

The utilization of processing resources at the NMS is the same as the WM approach and equal to (11).

3.5 Comparative Discussion

The precise trade-offs will depend on many parameters, but some general qualitative observations can be made which are mostly consistent with expectations. First, the polling rate Γ involves a basic trade-off between detection time and total traffic. The detection time decreases asymptotically to a fixed delay with larger Γ, but the total traffic increases linearly with Γ. With diminishing returns, there is no apparent reason to choose Γ greater than necessary to achieve a target detection time. Another possible criterion for choice of Γ might be a target sampling accuracy, e.g., the

probability of missing a variable change should be less than some target γ. A variable change may be missed if 2 or more changes occur within a sampling period of $1/\Gamma$. Because changes are a Poisson process, we have

$$\Pr(2 \text{ or more changes in } 1/\Gamma) = 1 - \left(1 + \frac{\lambda}{\Gamma}\right) e^{-\lambda/\Gamma} < \gamma \tag{17}$$

The sampling accuracy clearly increases as a function of Γ. Given Γ, the CS approach is not scalable to large networks because of the increasing traffic and processing burden on the NMS.

Second, the HS approach involves more traffic for routine monitoring than the CS approach. It also has longer delays for the NMS to detect changes. On the other hand, it reduces the traffic and processing burden on the centralized NMS as expected, which was the bottleneck in scaling the CS approach. Hence the HS approach is much more scalable to large networks than the CS approach. In the HS approach, the chosen number of L mid-level managers has no effect on the traffic or detection time, but the choice of L is constrained by the traffic and processing burden on each mid-level manager. Presumably, L should be small to minimize the cost of implementing mid-level managers, but large L will reduce the traffic and processing burden on each mid-level manager.

Third, the WM approach has desirable performance in terms of traffic and detection time, because the monitoring process takes place directly at the managed node and only variable changes are reported back to the NMS. There is a trade-off in the choice of sampling rate Γ; a faster sampling rate Γ will reduce the time to detect changes (asymptotically to a fixed delay) but consumes linearly increasing processing time. The selection of Γ will depend on the target detection time and constraints on processor utilization. Generally, this approach appears well suited for routine monitoring, but our analysis is limited to monitoring a single variable. If multiple variables are monitored, the memory and processing at each node would increase proportionally with the number of variables.

Finally, as may be expected, routine monitoring does not show the advantages of the SM approach. Indeed, all costs of this approach increase linearly with the number of agents L. It turns out that:

$$L = N\Gamma(t_s + t_p) \tag{18}$$

For this number of mobile agents, the mean detection time is equal to the CS and WM approaches, and the total traffic is $N(\Gamma\Pi_{MA} + \lambda I_r)$ bits/s which is comparable to the other approaches. However, the average memory usage per node is $\Gamma\Pi_{MA}t_s$, which will increase with Γ unlike the other approaches.

4. Searching for Data at an Unknown Node Location

One of the target applications often cited for mobile code is searching. We now consider a situation where specific data must be found by sequentially searching a random number of nodes. It is assumed that Q variables must be examined at each node; after searching at the n-th node, the search must continue to another node with probability $p(n)$ or is terminated with probability $1\text{-}p(n)$. This model can also cover the situation of searching for data at a known node location by letting $p(n) = 0$ for all n. If the node location is unknown and equally likely among the N nodes, as assumed here, then $p(n) = (N\text{-}n)/(N+1\text{-}n)$ and a search will cover $N/2$ nodes on average. The performance of each scheme is measured by the mean time to complete a search. The costs of interest are the amount of traffic (average capacity), average memory usage, and utilization of processing resources. We assume that I_q, I_r, t_p, t_c, t_q, and t_r have the same meaning as in routine network monitoring.

4.1 CS Model

Polling-based approaches are at a disadvantage for searching because all data must be fetched and processed by the network manager. An average search will take a total time:

$$T_{CS} = NQ(t_q + t_r + 2t_c + 2t_p)/2 \tag{19}$$

The average traffic handled by each node will be:

$$C_{CS} = Q(I_q + I_r)/2T_{CS} \text{ bits/s} \tag{20}$$

The total amount of traffic will be:

$$C_{CS} = NQ(I_q + I_r)/2T_{CS} \text{ bits/s} \tag{21}$$

The NMS handles the same total amount of traffic.

Again, agents reside permanently at each node so the memory consumption in each node is simply a fixed amount $M_{CS} = B$ bits.

The average utilization of processing resources per node is calculated as the processing time for messages divided by the mean search time:

$$U_{CS} = Qt_c / T_{CS} \tag{22}$$

Similarly, the NMS spends a total processing time $NQ(t_q + t_r)/2$ over the duration of the search, implying an average utilization of processing resources in the NMS:

$$U_{CS} = NQ(t_q + t_r)/2T_{CS} \tag{23}$$

4.2 HS Model

In the HS approach, it is possible to exploit the L mid-level managers to carry out searches in parallel. When the data is found, the final results are reported to the high-level NMS in a response message. It is assumed that all mid-level managers will be able to terminate their searches when the data is found. The mean time to complete the search is:

$$T_{HS} = \frac{NQ(t_q + t_r + 2t_c + 2t_p)}{2L} + 2(t_q + t_r + t_p) \tag{24}$$

This is approximately shorter than (19) by a factor of L.

The traffic capacity of each node is:

$$C_{HS} = Q(I_q + I_r) / 2T_{HS} \text{ bits/s} \tag{25}$$

The traffic handled by each mid-level manager is approximately

$$C_{HS} \cong NQ(I_q + I_r) / 2LT_{HS} \text{ bits/s} \tag{26}$$

ignoring the initial query message from the high-level NMS. The NMS sends L query messages and receives one response message over the duration of the search:

$$C_{HS} = (LI_q + I_r) / T_{HS} \tag{27}$$

The hierarachy does not change the memory usage $M_{HS} = B$ bits at each node.

The utilization of processing resources at each node is increased from (22) to:

$$U_{HS} = Qt_c / T_{HS} \tag{28}$$

The fraction of total computing time spent by each mid-level manager is:

$$U_{HS} = NQ(t_q + t_r) / 2LT_{HS} \tag{29}$$

except for the mid-level manager that found the data will spend slightly more time to generate a message to the NMS. The processing burden on the high-level NMS is:

$$U_{HS} = (Lt_q + t_r) / T_{HS} \tag{30}$$

4.3 WM Model

The WM approach differs from the CS approach by sending the agent code to a node to examine Q variables there and reporting back a result at the end of the task. For

speed, code may be dispached to L nodes in parallel. On average, $N/2$ nodes will be searched. The expected time to complete the search is:

$$T_{WM} = N(Qt_s + t_q + t_r + 2t_c + 2t_p) / 2L \tag{31}$$

The average traffic handled by each node is

$$C_{WM} = (I_{WM} + I_r) / 2T_{WM} \tag{32}$$

The total traffic in the network is

$$C_{WM} = N(I_{WM} + I_r) / 2T_{WM} \tag{33}$$

The NMS handles the same total amount of traffic.

For memory usage, a dispatched code resides on a node for a time Qt_s. The average memory usage per node is

$$M_{WM} = Qt_s I_{WM} / 2T_{WM} \tag{34}$$

The utilization of processing resources at each node is

$$U_{WM} = (Qt_s + 2t_c) / 2T_{WM} \tag{35}$$

The utilization of processing resources at the NMS is

$$U_{WM} = N(t_q + t_r) / 2T_{WM} \tag{36}$$

4.4 SM Model

In the SM model, the NMS initially dispatches L mobile agents that search the nodes sequentially. When the data is found, the agent reports the results back to the NMS, and all agents will be terminated. This approach reduces the processing burden on the NMS but the state carried by the mobile agent may increase with each node.

The expected time to complete the search is

$$T_{MA} = \frac{N(Qt_s + t_p)}{2L} + t_q + t_r + t_c + 2t_p \tag{37}$$

The traffic handled by a node divided by the mean search time is

$$C_{MA} = (4LI_{MA} + aN) / 4LT_{MA} \tag{38}$$

The total traffic in the network is

$$C_{MA} = N(4LI_{MA} + aN) / 8LT_{MA} \tag{39}$$

The NMS handles only initial transmission of L agents and receipt of the final response message:

$$C_{MA} = (LI_{MA} + I_r) / T_{MA} \tag{40}$$

For memory usage, the average memory usage per node is

$$M_{MA} = Qt_s(4LI_{MA} + aN) / 8LT_{MA} \tag{41}$$

The utilization of processing resources at each node is:

$$U_{MA} = Qt_s / 2T_{MA} \tag{42}$$

The utilization of processing resources at the NMS is

$$U_{MA} = (Lt_q + t_r) / T_{MA} \tag{43}$$

4.5 Comparative Discussion

We attempt general observations that are independent of the particular parameter values. First, the CS approach is not scalable because the search time is proportional to the size of the network. This centralized approach cannot take advantage of the parallelism in the other approaches. The search time for the other approaches can all be reduced by a factor of L, the degree of parallelism.

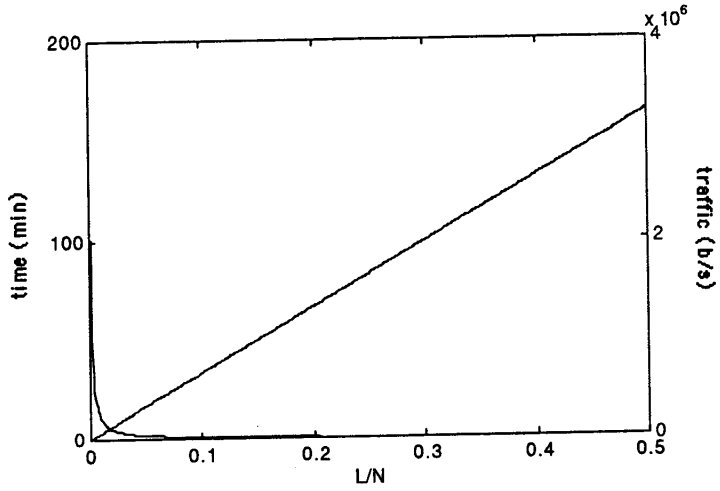

Figure 2. Search time and total traffic for HS approach with $I_q = I_r = 100$ bytes, $Q = 50$, $t_q = t_r = t_c = 10$ ms, $t_p = 100$ ms, $N = 1000$.

On the other hand, parallelism will increase the total amount of traffic. Figure 2 shows that the search time (left axis) decreases rapidly with L, but the total traffic (right axis) increases approximately linearly with L. Hence, there is diminishing returns with too many mid-level managers. The WM approach exhibits a similar

trade-off. The detection time decreases rapidly with L, but the total traffic increases linearly.

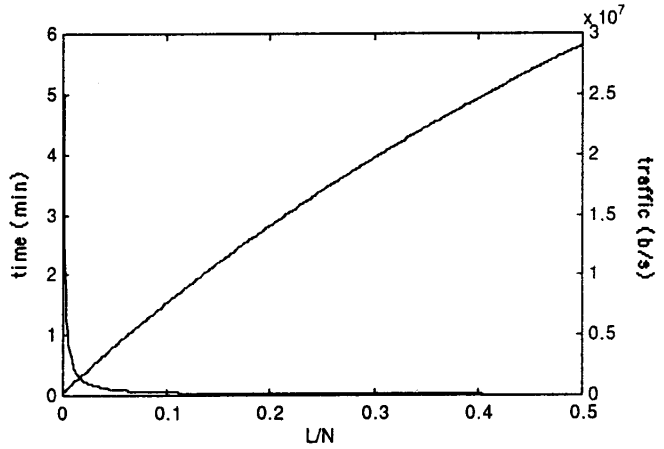

Figure 3: Search time and total traffic for SM approach with a = 500 bits, I_{MA} = 6000 bytes, Q =50, $t_q = t_r = t_c = 10$ ms, t_p=100 ms, t_s=0.01 s, N = 1000.

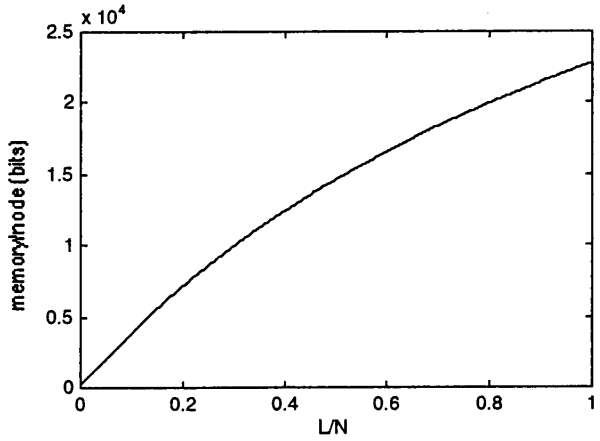

Figure 4: Memory usage per node for SM approach with a = 500 bits, I_{MA} = 6000 bytes, Q = 50, $t_q = t_r = t_c = 10$ ms, $t_p = 100$ ms, $t_s = 0.01$ s, N = 1000.

The SM approach appears favorable for the searching application. Figure 3 shows that the total traffic for the SM approach grows less than linearly with L, although the search time decreases rapidly with L. The reason is because the mobile agents tend to be smaller size on average when the search time is shorter. As L increases, there are more agents but the average agent is smaller.

In other aspects as well, the SM approach compares well with the other approaches. In terms of the traffic and processing burden on the NMS, both SM and HS approaches are proportional to L and more scalable than the CS and WM approaches.

Because the SM approach is entirely distributed among the nodes, the memory usage in the nodes increases with L. That is, the SM approach may impose significant processing and memory burden on the nodes. However, Figure 4 shows that the memory usage is less than linear with L.

5. Conclusions

As might be expected, distributed approaches can have major advantages over traditional centralized network management. However, a single distributed approach may not be the best for all applications. With the models and analysis here, it appears that the weak mobility approach performs well for routine monitoring, and the mobile agent approach is well suited for searching types of applications. However, the trade-offs between the different approaches need careful quantification before the best approach can be determined for a specific application.

The relatively simplistic models developed here may be improved in several ways. For example, the representation of memory and computing resources might be more realistic. Also, the migration paths of mobile agents could be more complex.

References

[1] J. Case, M. Fedor, M. Schoffstall, J. Davin, "A simple network management protocol (SNMP)," Internet RFC 1157, 1990.

[2] M. Siegl, G. Trausmuth, "Hierarchical network management: a concept and its prototype in SNMPv2," *Comp. Net. & ISDN Sys.*, vol. 28, pp. 441-452, April 1996.

[3] G. Goldszmidt, Y. Yemini, "Delegated agents for network management," *IEEE Commun. Mag.*, vol. 36, pp. 66-70, March 1998.

[4] A. Bieszczad, B. Pagurek, T. White, "Mobile agents for network management," *IEEE Commun. Surveys*, vol. 1, 4Q1998, http://www.comsoc.org/pubs/surveys.

[5] M. Kahani, H. Beadle, "Decentralized approaches for network management," *Computer Commun. Rev.*, vol. 27, pp. 36-47, July 1997.

[6] M. Baldi, S. Gai, G. Picco, "Exploiting code mobility in decentralized and flexible network management," *Int. Workshop on Mobile Agents*, April 1997.

[7] A. Fuggetta, G. Picco, G. Vigna, "Understanding code mobility," *IEEE Trans. Software Engin.*, vol. 24, pp. 342-361, May 1998.

[8] A. Carzaniga, G. Picco, G. Vigna, "Designing distributed applications with mobile code paradigms," *Int. Conf. on Software Engin.*, pp. 22-32, 1997.

[9] M. Baldi, G. Picco, "Evaluating the tradeoffs of mobile code design paradigms in network management applications," *ICSE'98*, pp. 146-155, April 1998.

[10] A. Liotta, G. Knight, G. Pavlou, "Modelling network and system monitoring over the Internet with mobile agents," *NOMS'98*, pp. 303-312, Feb. 1998.