A Model and Evaluation of Distributed Network Management Approaches

Thomas M. Chen, *Senior Member, IEEE* and Stephen S. Liu, *Senior Member, IEEE*

***Abstract*—This paper develops an analytical framework to model and compare centralized and distributed approaches for network management. Each scheme is evaluated in terms of performance and scalability for two applications, network monitoring, and data searching. The results support the intuitive argument that distributed approaches can have considerable advantages over traditional centralized network management, but a single approach may not be best for all types of applications. Instead, the most appropriate approach for a specific application should be selected after a careful evaluation. The modeling framework presented in this paper is intended to quantify the tradeoffs between different approaches to lend a basis for the selection decision.**

***Index Terms*—Distributed network management, mobile agents.**

I. Introduction

CURRENTLY, DATA networks are managed mainly by simple network management protocol (SNMP) and telecommunications networks by common management information protocol (CMIP) which are both client-server approaches [1], [2]. Centralized network management systems (NMSs) are clients to management agents residing permanently in each managed network element (NE). Although adequate for most practical management applications, the limitations of SNMP—for example, the potential processing and traffic bottleneck at the NMS—have been recognized for many years.

It has been observed that decentralizing network management functions may achieve several benefits [3]–[5]. Distributed network management offers several perceived advantages: network traffic and processing load in the NMS can be both reduced by performing data processing closer to the NEs; scalability to large networks is improved; searches can be performed closer to the data, improving speed and efficiency; and distributed network management is inherently more robust without depending on continuous communications between the NMS and NEs. Recent attention to distributed network management has increased because the processing capabilities of routers and switches have improved considerably and the popularization of Java and CORBA have brought mobile code concepts closer to mainstream acceptance [6], [7].

Despite the perceived benefits however, commercial deployment has been slow for various reasons. The main issue has been concerns about security and safety [8]. Second, a common platform is required to support code mobility [9]. Although various mobile code systems have been developed over the years, none has been widely deployed except for the Java virtual machine and interoperability standards such as Mobile Agent System Interoperability Facility (MASIF) have been started only recently [10], [11]. Third, misperceptions have often been associated with mobile code systems. For example, contrary to some claims, mobile code systems are usually intended to augment traditional network management systems, not to replace them. Not every management application warrants the additional complexity and costs (for security, safety, agent mobility, coordination).

Unfortunately, quantitative studies of decentralized network management approaches have been infrequent and limited [12]–[15]. The objective of analysis is not necessarily to uncover new insights; a quantitative treatment can lend support to previously known but intuitively argued tradeoffs. However, it is difficult to model all complexities of a managed network and any model will be an approximation to a real system. This paper attempts to present a more comprehensive model and evaluation than previous studies in a few respects: costs considered here include more than traffic; the network may be nonuniform (distances to each NE may differ); the existence of security and safety mechanisms is recognized; and different management applications are examined. In Section II, we develop the basic models and assumptions for four prototypical management approaches. In Section III, the four approaches are evaluated for a routine monitoring application to detect a change in network status. In Section IV, the different approaches are analyzed for a searching application. Section V presents a comparative discussion of the results.

II. Distributed Network Management

A. Approaches

Centralized network management is inefficient because every management action depends on the NMS. Clearly, the inefficiency can be reduced by distributing network management functions to a hierarchy of midlevel managers, each responsible for managing a portion of the entire network, as in SNMPv2 [16]–[18]. Subnetworks can be managed in parallel, reducing the traffic and processing burden on the highest level NMS. Typically, the distribution of network management functions and the organization of managers are fairly static.

In the management by delegation (MBD) approach, management functions may be distributed dynamically by dispatching "delegated agents" to a managed NE where the agent code is executed [3], [19]. The NEs include an "elastic process" that can

Manuscript received March 20, 2001; revised November 30, 2001.

T. M. Chen is with the Department of Electrical Engineering, Southern Methodist University, Dallas, TX 75275 USA (e-mail: tchen@engr.smu.edu).

S. S. Liu is with the Verizon Laboratories, Inc., Waltham, MA 02254 USA (e-mail: steve.liu@verizon.com).

Publisher Item Identifier S 0733-8716(02)03064-0.

dynamically add functions (patches) carried by delegated agents and delete them after a function is carried out. This dynamic approach offers the advantage of instantiating management functions (e.g., delegating agent code) only when they are needed, but involves a cost of additional complexity. Hence, MBD was proposed only for certain applications where the cost was justified by the need to reduce processing delay or network traffic [19].

MBD is an example of "weak mobility" where delegated agents do not migrate after execution at a NE [12]. Weak mobility can be implemented by two basic methods: remote evaluation (REV) or code on demand (COD) [5], [13]. In the REV method, the code for the delegated task is sent directly from the manager to the NE where it is executed [20]. In the COD method, a task but not the code is delegated to the managed node (the necessary code is fetched from the network, possibly a trusted code server).

In contrast, a "strong mobility" capability allows an agent to suspend its execution at one NE, transfer its code and execution state (and perhaps data) to another NE and resume execution there. This is usually referred to as a mobile agent although this term is used inconsistently in the literature [21]. The itinerary of a mobile agent may depend on the data it finds, so 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.

B. Models and Assumptions

Previous studies have concentrated on traffic costs assuming a uniform network (equidistant nodes) and negligible computing costs. Carzaniga, Picco, and Vigna identified four approaches: client-server, remote evaluation, code on demand, and mobile agent [13]. For a uniform network, the approaches were compared in terms of overhead traffic for a data mining application. Baldi and Picco also evaluated the same approaches for an example network monitoring application in terms of total traffic and the traffic concentrated at the NMS [14]. Mobile agents were found to be always disadvantageous and remote evaluation was better than client-server under certain conditions. A similar study is reported by Fuggetta, Picco, and Vigna [12]. Liotta, Knight, and Pavlou considered a mobile agent architecture (only weak mobility) with a hierarchy of network managers and modeled traffic (bandwidth) costs for a monitoring application as a function of monitoring duration [15]. Our study examines a slightly different classification of approaches for a nonuniform network and attempts to include additional cost metrics such as delays, computing resources, and security measures in the model. To discern the relative strengths of each approach, we consider two scenarios: network monitoring and data searching.

The network management approaches considered here are shown in Fig. 1. In Fig. 1(a), the *client-server (CS)* model represents the traditional SNMP paradigm where a centralized NMS polls a network of N network elements. The communications between the NMS and agents is characterized by pairs of query-response messages for every interaction. Fig. 1(b) represents a *hierarchical static (HS)* approach modeled as L midlevel managers, each managing a separate subnetwork of N/L network elements. A two-level hierarchy is considered here although multiple hierarchical layers may evidently be possible. Each subnetwork is managed in a client-server manner and midlevel managers may communicate with a centralized high-level NMS as needed.

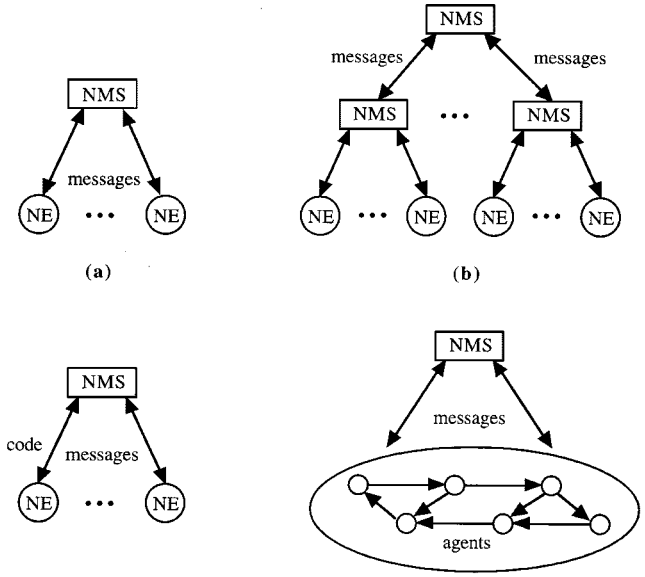

Fig. 1. Centralized and distributed network management approaches.(a) Client-server. (b) Hierarchical static. (c) Weak mobility. (d) Strong mobility.

In the *weak mobility (WM)* approach, the NMS distributes code to specific NEs where the code is executed, as shown in Fig. 1(c). After performing its specific task, the code will typically report results back to the NMS and expire at the NE. During execution, the code does not have a capability for autonomous migration to other NEs. In the *strong mobility (SM)* approach, the NMS dispatches one or more agents to carry out a specific task, as shown in Fig. 1(d). Agents have the capability to autonomously travel (execution state and code) among different NEs to complete its task. The route may be predetermined or chosen dynamically depending on the results at each NE.

Generally, a performance metric of interest will be the time to complete a specific task. To model nonuniform delays through the network, we represent packet delay by a parameterized random variable $t_p(N)$ which is dependent on the size of the network, N nodes. The average packet delay is denoted by $\bar{t}_p(N)$. Packet processing times are generally assumed to be constants, although they may be variable and effected by processor load in actuality. We will also be concerned with the scalability of each approach in terms of costs in traffic (at the NMS and NEs), memory resources and average utilization of processing resources.

The CS approach represents a network being managed within a single administrative domain and the need for security is assumed to be minimal. In the HS approach with multiple midlevel managers, packets may need to be authenticated and encrypted against unauthorized access to management data. The authentication and encryption mechanisms in the SNMPv3 user-based security model are assumed to be sufficient in this case [22]. These mechanisms involve the computation of MD5 or SHA-1 message digests for authentication and encryption by DES. The additional security is factored as longer packet processing times and additional message overhead.

Many current mobile code systems are based on Java, therefore, we examine the Java security model for the WM and SM approaches [8], [23], [24]. The original Java Development Kit (JDK) security model depended on the sandbox model consisting of byte-code verification, a class loader to install each applet in a separate name space and a security manager to mediate run-time access to system resources. JDK 1.1 introduced digital signatures to allow correctly signed remote code to be treated as trusted local code. JDK 1.2 enabled fine-grained access control based on individual access permissions. A new Java Authentication and Authorization Sevice allows authentication by a variety of credentials such as passwords, Kerberos tickets, or public key certificates [25]. As an example, we might assume additional time will be required to fetch and process certificates and certificates will increase bandwidth requirements.

III. A Network Monitoring Application

We first consider the problem of detecting a random change in a variable X at an arbitrary NE. Changes in the variable are assumed to occur randomly as a Poisson process with rate λ (i.e., times between changes are independent exponential random variables with mean $1/\lambda$) at any NE. The main performance metric of interest is the mean time T to detect the change. The costs of interest include:

- C amount of traffic measured as capacity (b/s) at the NMS and NEs;
- M average amount of memory (bits) consumed at NEs;
- U average utilization of processing resources at the NMS and NEs.

A. CS Approach

In the CS approach, the NMS will regularly poll each NE to retrieve the value of variable X. The CS approach is characterized by these variables:

- Δ polling interval between consecutive requests;
- i_q size of each query message (bits);
- i_r size of each response message (bits);
- h size of packet header (bits);
- t_c processing time for a query or response message at a NE;
- t_q processing time for a query message at NMS;
- t_r: processing time for a response message at NMS.

Within any polling interval of Δ, the Poisson process implies that a variable change is likely to occur at any time uniformly distributed within the interval. The average time until the next query is received at the NE after a variable change is $\Delta/2$. After receiving the query, the NE will transmit a response message to the NMS. Including all processing times, the expected time for the NMS to detect a variable change is the sum

$$T_{CS} = \frac{\Delta}{2} + 2t_c + t_r + \bar{t}_p(N). \tag{1}$$

Each poll consists of a pair of query-response messages, so each NE will handle $(i_q+i_r+2h)/\Delta$ b/s. The NMS must handle the total traffic from N nodes

$$C_{CS} = \frac{N\,(i_q + i_r + 2h)}{\Delta} \text{ b/s}. \tag{2}$$

The traffic bottleneck at the NMS is evident from the linear dependence on N.

A simple agent resides permanently at each NE and memory consumption is not an issue. The agent spends a time t_c to process a query or response message in every polling interval Δ. The fraction of time spent by each NE on packet processing is $2t_c/\Delta$. Of greater concern is the fraction of time spent by the NMS on packet processing which is

$$U_{CS} = \frac{N\,(t_q + t_r)}{\Delta}. \tag{3}$$

The processing bottleneck at the NMS is obvious because the NMS will be completely occupied when the network size is $N \geq \Delta/(t_q + t_r)$.

B. HS Approach

In the HS model, the monitoring function is delegated from the high-level NMS to L midlevel managers which simultaneously manage subnetworks of N/L nodes in client-server fashion.The costs for encryption and authentication are represented by an additional packet processing delay t_s at sender and receiver and an overhead per packet of h_s b. The additional security delay t_s may not be substantial compared with other delays because encryption techniques and equipment are common. With the same polling rate per node, the expected time for a midlevel manager to detect a variable change is

$$T_{HS} = \frac{\Delta}{2} + 2t_c + t_r + 3t_s + \bar{t}_p\left(\frac{N}{L}\right) \tag{4}$$

which is better than (1) if $3t_s + \bar{t}_p(N/L) < \bar{t}_p(N)$. We must add the additional time $t_r + 2t_s + \bar{t}_p(N)$ 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

$$T_{HS} = \frac{\Delta}{2} + 2t_c + 2t_r + 5t_s + \bar{t}_p\left(\frac{N}{L}\right) + \bar{t}_p(N). \tag{5}$$

The detection time is longer than (1) due to the additional security and longer route through the midlevel manager.

With the same polling interval Δ, the operation of each NE is not drastically different from the CS case. Each NE will handle $(i_q + i_r + 2h + 2h_s)/\Delta$ b/s, a slight increase due to security overhead. Each midlevel manager must handle $N(i_q + i_r + 2h + 2h_s)/L\Delta$ b/s between the N/L network elements being monitored and send $N\lambda(i_r + h + h_s)/L$ b/s on average to the high-level NMS if all variable changes are detected. The midlevel managers are beneficial if they perform some data filtering and report only variable changes to the high-level NMS. The NMS is relieved of sending query messages and just receives a total

$$C_{HS} = N\lambda\,(i_r + h + h_s) \text{ b/s} \tag{6}$$

of response messages. Although the traffic (6) is still linearly dependent on N, it should be less than the CS case (2) if the polling rate $1/\Delta$ is greater than the variable change rate λ.

Processing resources are a more important issue than memory consumption (which is constant as in the CS case). Each midlevel manager spends a total time $N(t_q+(1+\lambda\Delta)(t_r+2t_s))/L$ on packet processing in every polling interval Δ, or fraction of time $N(t_q+(1+\lambda\Delta)(t_r+2t_s))/L\Delta$. For the midlevel managers to avoid being completely occupied with packet processing, we need at least $L > N(t_q + (1 + \lambda\Delta)(t_r + 2t_s))/\Delta$ midlevel

managers. The fraction of total computing time spent by the high-level NMS is

$$U_{HS} = N\lambda(t_q + t_s) \tag{7}$$

which is unavoidably dependent on the total rate of variable changes in the network. The processing bottleneck at the NMS is partially relieved and the NMS can cover up to $N = 1/\lambda(t_q+t_s)$ nodes.

C. WM Approach

In the WM approach, the code for network monitoring is dispatched initially from the NMS to all NEs, fetched from trusted code servers, or copied from NE to NE. For routine network monitoring, there is no need to dispatch code dynamically and this advantage of the WM approach will be imperceptible. The NEs may be programmed to report the variable X value at periodic intervals of Δ or more intelligently report only variable changes. The former approach will reduce the traffic compared with the CS case by eliminating the need for query messages, while the latter approach will reduce the traffic further to simply

$$C_{WM} = N\lambda(i_q + h + h_s) \text{ b/s} \tag{8}$$

to the NMS. We have assumed the same level of security as the HS approach because only messages and not program code are exchanged.

The agents may be programmed to detect variable changes immediately or "sample" the current variable value at periodic intervals of Δ. The latter approach is assumed because a continuous process might consume excessive computing resources. After a variable change, the average time until the next sample will be $\Delta/2$. Including the time to generate and transmit a response message to the NMS, the mean time to detect a variable change will be

$$T_{WM} = \frac{\Delta}{2} + t_r + t_c + 2t_s + \bar{t}_p(N). \tag{9}$$

If agents sample the variable periodically, then processing time and memory consumption at the NEs are not an issue. The processing time at the NMS is more interesting and will depend on whether agents report periodically or only variable changes. If agents report periodically, the NMS may become a processing bottleneck as in the CS case. If agents report only variable changes, the fraction of processing time spent by the NMS on packet processing will be the same as (7).

D. SM Approach

Mobile agents can monitor a number of nodes in turn, reducing the amount of traffic between the NMS and NEs compared with the CS approach. Compared with static approaches, mobile agents will not continuously consume memory resources. For routine monitoring, we assume that L mobile agents each travel around a separate group of N/L network elements in a preset pattern. Each agent dwells for a time t_a at each node (long enough to sample variable X) and moves to the next node with forwarding delay $t_p(N/L)$. Each move must be associated with an additional cost to authenticate and install the agent at the next NE. This cost could be substantial involving messages and computing time, but depends a great deal on the particular system [26]. For generality, we represent the additional cost as a random variable t_{ma} with mean $\bar{t}_{ma}$. If a variable change is detected, the agent will immediately send a response message to the NMS. The time for an agent to visit each NE is $(t_a + \bar{t}_p(N/L) + \bar{t}_{ma})$, so the effective polling interval per NE is $N(t_a + \bar{t}_p(N/L) + \bar{t}_{ma})/L$. Including the time to generate and transmit a secure response message to the NMS, the mean time to detect a variable change will be

$$T_{SM} = \frac{N\left(t_a + \bar{t}_p\left(\frac{N}{L}\right) + \bar{t}_{ma}\right)}{2L} + t_r + t_c + 2t_s + \bar{t}_p(N). \tag{10}$$

Although mobile agents may generally migrate state as well as code, the capability to accumulate data is not needed for the routine monitoring application under consideration. Thus, agents are assumed to be a fixed size i_{ma} b including code and certificate. Each NE must handle the receipt and forwarding of i_{ma} b in every polling interval, or a total traffic of

$$C_{SM} = \frac{2Li_{ma}}{N\left(t_a + \bar{t}_p\left(\frac{N}{L}\right) + \bar{t}_{ma}\right)} \text{ b/s}. \tag{11}$$

Assuming that agents report only variable changes, the NMS handles the same traffic as the WM case (8).

Unlike the weak mobility approach, mobile agents will consume memory only during its dwelling time t_a in each polling interval. Hence, the average memory usage per node is

$$M_{SM} = \frac{Li_{ma}t_a}{N\left(t_a + \bar{t}_p\left(\frac{N}{L}\right) + \bar{t}_{ma}\right)} \text{ b}. \tag{12}$$

Finally, considering the time to authenticate and install each agent, each NE spends a processing time $(t_a + \bar{t}_{ma})$ in each polling interval, implying an average utilization of processing resources

$$U_{SM} = \frac{L(t_a + \bar{t}_{ma})}{N\left(t_a + \bar{t}_p\left(\frac{N}{L}\right) + \bar{t}_{ma}\right)}. \tag{13}$$

The utilization of processing resources at the NMS is the same as the WM and HS cases (7).

IV. A Data Searching Application

Searching is often cited as one of the primary applications for mobile code. We now consider a situation where specific data must be found by sequentially searching a number of NEs. It is assumed that Q variables must be examined at each NE; after searching at the nth NE, the search must continue to another NE with probability $p(n)$ or can be terminated with probability $1 - p(n)$. We assume that the CS approach must perform the search sequentially, whereas the other decentralized approaches are capable of a certain degree of parallelism. In the decentralized approaches, L searches are performed simultaneously in disjoint groups of N/L network elements. As shown in Fig. 2, this is a fairly general navigation model that can cover the special cases that the target data resides at a known NE (by letting $L = 1$ and $p(n) = 0$ for all n), all NEs must be searched sequentially (by letting $L = 1$), or all NEs are searched at the same time (by letting $L = N$). If the data location is unknown and equally likely among the N network elements, as assumed here, then $p(n) = (N-n)/(N+1-n)$ and a search will cover $N/2$ network elements on average.

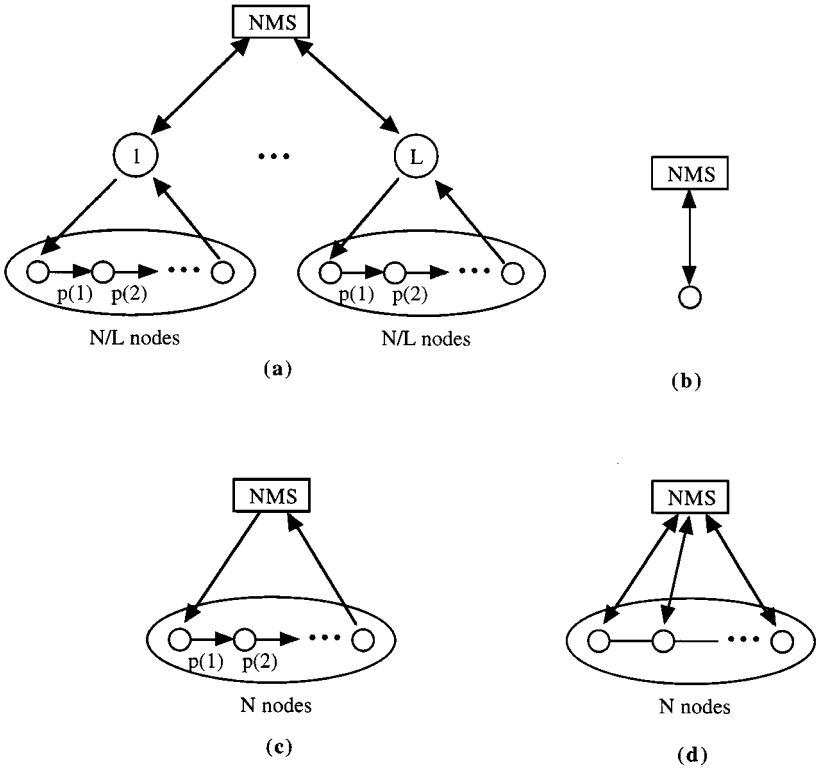

Fig. 2. Search patterns. (a) General pattern with L parallel searches and continuation probability $p(n)$. (b) Special case with $L = 1$ and $p(n) = 0$ for all n. (c) special case with $L = 1$. (d) Special Case with $L = N$.

A. CS Approach

The centralized polling approach is clearly disadvantageous for searching because all data must be fetched and processed by the NMS. We assume that a "bulk query" message (as in SNMPv2) can retrieve a block of Q variables in a single bulk-response message. Because the bulk-response message is now larger, the bulk-response message is $Qi_q + h$ b and requires a processing time Qt_c and Qt_r at the NE and NMS, respectively. Each query-response message pair involves a total time of $(t_q + Qt_r + (Q+1)t_c + 2\bar{t}_p(N))$ including packet processing times and forwarding delays. An average search will take a total time of

$$T_{CS} = \frac{N\left(t_q + Qt_r + (Q+1)t_c + 2\bar{t}_p(N)\right)}{2}. \tag{14}$$

The total traffic in an average search will consist of $N/2$ query-response message pairs or $N(i_q + Qi_r + 2h)/2$ b. Divided by the average search duration, the total amount of traffic through the NMS will be

$$C_{CS} = \frac{i_q + Qi_r + 2h}{t_q + Qt_r + (Q+1)t_c + 2\bar{t}_p(N)} \text{ b/s}. \tag{15}$$

The processing burden on the NMS is a concern because the search is conducted centrally. The NMS spends a processing time $(t_q + Qt_r)$ for each query-response message pair, implying an average fraction of time spent on packet processing

$$U_{CS} = \frac{t_q + Qt_r}{t_q + Qt_r + (Q+1)t_c + 2\bar{t}_p(N)}. \tag{16}$$

B. HS Approach

In the HS approach, it would be advantageous to exploit the L midlevel managers to carry out searches in parallel, or otherwise the search time will be similar to the CS case. The high-level NMS can distribute an initial query message to all midlevel managers. When the data is found, the final results are reported to the high-level NMS in a response message which will then terminate all searches with an appropriate message to all midlevel managers. The mean time to complete the entire search is the sum of searching time by a midlevel manager and time to report from the midlevel manager to the NMS

$$T_{HS} = \frac{N\left(t_q + Qt_r + (Q+1)t_c + 2\bar{t}_p\left(\frac{N}{L}\right)\right)}{2L} + t_c + t_r + 2t_s + \bar{t}_p(N). \tag{17}$$

While the search time by a midlevel manager over N/L nodes is shorter than (14), there is an additional delay due to the response message reporting the search result to the NMS.

The total traffic in an average search will be unchanged (i.e., each midlevel manager will handle $N/2L$ query-response message pairs on average). Instead of all traffic going through the NMS, the management hierarchy will shift the traffic load to the midlevel managers. The NMS handles only the initial query messages and the final response message with the search results. Divided by the average search time, the traffic through each midlevel manager will be

$$C_{HS} = \frac{i_q + Qi_r + 2h}{t_q + Qt_r + (Q+1)t_c + 2\bar{t}_p\left(\frac{N}{L}\right)} \text{ b/s}. \tag{18}$$

The traffic is slightly higher than the CS case (15) because the search areas are N/L instead of N network elements and the packet forwarding time $\bar{t}_p(N/L)$ is shorter than $\bar{t}_p(N)$.

The management hierarchy will also reduce the computational burden on the NMS to processing the final response message. Instead, the processing burden will be shifted to the midlevel managers which will spend an average fraction of time on packet processing

$$U_{HS} = \frac{t_q + Qt_r}{t_q + Qt_r + (Q+1)t_c + 2\bar{t}_p\left(\frac{N}{L}\right)}. \tag{19}$$

The processing is more intense than the CS case (16), again due to the smaller search areas and shorter packet forwarding time.

C. WM Approach

In the WM model, agent code is dispatched to an NE to examine Q variables and report back the search result. Agents have no autonomous migration capability and the search is controlled centrally requiring continual communications with the NMS. For efficiency, L agents may be dispatched simultaneously. A search by these L agents involves a time to dispatch an agent of size i_{wm} b consisting of code and certificate, authenticate and install the code, perform the search of Q variables and report the results to the NMS in a secure response message. The time to dispatch an agent will be denoted by t_{wm}. For simplicity, the time to authenticate and install the code will be assumed to be the same random variable t_{ma} used earlier for the SM case. The time to search Q variables is Qt_a where t_a was used earlier to denote the time to sample one variable value. Including packet forwarding times, the total expected time to complete the search will be $(t_{wm} + \bar{t}_{ma} + Qt_a + t_c + t_r + 2t_s + 2\bar{t}_p(N))$. On average, this search may have to be repeated $N/2L$ times before the target data is found. Hence, the expected time to complete the entire search will be

$$T_{WM} = \frac{N\left(t_{wm} + \bar{t}_{ma} + Qt_a + t_c + t_r + 2t_s + 2\bar{t}_p(N)\right)}{2L}. \tag{20}$$

The centralized control raises concern about a possible traffic bottleneck at the NMS. The NMS sends L agents and receives L response messages within an average search duration, implying an average traffic flow of

$$C_{WM} = \frac{L\left(i_{wm} + i_r + h + h_s\right)}{t_{wm} + \bar{t}_{ma} + Qt_a + t_c + t_r + 2t_s + 2\bar{t}_p(N)} \text{ b/s}. \tag{21}$$

The memory consumption and processing times at the NEs are not a major concern because agents visit each NE for a fraction of the total search time. Of greater concern may be the processing burden at the NMS. The NMS spends processing time to dispatch L agents and process L response messages within an average search time. The fraction of time spent on processing by the NMS will be

$$U_{WM} = \frac{L\left(t_{wm} + t_r + t_s\right)}{t_{wm} + \bar{t}_{ma} + Qt_a + t_c + t_r + 2t_s + 2\bar{t}_p(N)}. \tag{22}$$

D. SM Approach

In the SM model, the NMS will dispatch L mobile agents that will each search through subnetworks of N/L network elements. On average, $N/2L$ network elements will be visited by each agent before the target data is found. When the data is found, the agent will report the search results back to the NMS. For efficiency, the infrastructure must support means for agent coordination to terminate all searches after the data is found. This approach reduces the processing burden on the NMS but the state carried by the mobile agent may increase with each node because agents retain some data about their completed searches. In addition to i_{ma} b of code and certificate, a mobile agent may carry $s(n)$ b of state after visiting the nth NE. For simplicity, the state is assumed to be a linear function $s(n) = an$.

The time for an agent to visit each NE is $(Qt_a + \bar{t}_p(N/L) + \bar{t}_{ma})$ consisting of a time to search Q variables, migrate to the next NE and authenticate and install itself at the next NE. This ignores the effect of the growing state on the forwarding delay. Since each agent visits $N/2L$ network elements on average, the total expected search time including a secure response message to the NMS will be

$$T_{SM} = \frac{N\left(Qt_a + \bar{t}_p\left(\frac{N}{L}\right) + \bar{t}_{ma}\right)}{2L} + t_c + t_r + 2t_s + \bar{t}_p(N). \tag{23}$$

In this distributed approach, the NMS is relieved of traffic and processing. Of greater concern is the traffic and processing burden at NEs. Since an agent visits $N/2L$ nodes, the average agent size will be approximately $i_{ma} + aN^2/8L^2$ b. The average traffic handled by each NE will be

$$C_{SM} = \frac{8L^2 i_{ma} + aN^2}{4L^2\left(Qt_a + \bar{t}_p\left(\frac{N}{L}\right) + \bar{t}_{ma}\right)} \text{ b/s}. \tag{24}$$

For memory consumption, a mobile agent will reside at a NE for a time Qt_a. On average, $N/2$ nodes are visited; divided by N nodes and the mean search time, the average memory consumption per NE will be

$$M_{SM} = \frac{Qt_a\left(8L^2 i_{ma} + aN^2\right)}{8L\left(\left(Qt_a + \bar{t}_p\left(\frac{N}{L}\right) + \bar{t}_{ma}\right) + 2L\left(t_c + t_r + 2t_s + \bar{t}_p(N)\right)\right)} \tag{25}$$

By a similar analysis considering that an agent will spend a time Qt_a at a NE during each search, the average fraction of time by each NE on processing will be

$$U_{SM} = \frac{Qt_a L}{N\left(Qt_a + \bar{t}_p\left(\frac{N}{L}\right) + \bar{t}_{ma}\right) + 2L\left(t_c + t_r + 2t_s + \bar{t}_p(N)\right)} \tag{26}$$

V. Comparative Discussion

A. Network Monitoring

The limited scalability of the CS approach for network monitoring is obvious from the linear dependence of traffic and processing utilization at the NMS on the network size N. We found that the processing bottleneck limits the network size to $N = \Delta/(t_q + t_r)$.

To compare the HS approach, the packet forwarding time $\bar{t}_p(N)$ might be considered as proportional to the diameter of a circle with an area of N. Reducing the circular area to N/L will reduce the diameter by a factor of $\sqrt{L}$. Hence, we can consider a rough approximation $\bar{t}_p(N/L) \approx \bar{t}_p(N)/\sqrt{L}$.

In the HS approach, the number of midlevel managers, L, is clearly an important parameter effecting performance. There is a minimum value of L to avoid a processing bottleneck at the midlevel managers depending on N and Δ. The HS approach appears to be much more scalable than the CS approach, if L can be chosen sufficiently large, but in practice the improved performance must be weighed against a cost for additional midlevel managers.

The WM approach exhibits favorable performance capable of minimizing the detection time and traffic, under the assumption that agents are programmed with the intelligence to report only variable changes. However, we have not exploited the capability of code mobility in this scenario.

The performance of the SM approach in (10) is very dependent on the dwelling time per node, t_a and the time to authenticate and install agents, $\bar{t}_{ma}$. It is likely that $\bar{t}_{ma}$ could be a significant factor because mobile agents will require strong security measures for authentication. Performance can be improved to some extent by increasing L but this will linearly increase the traffic and processing burden on each NE.

B. Searching for Data

The limitations of the CS approach are obvious for a searching application because the search is conducted sequentially and cannot take advantage of parallelism unlike the other approaches. The search time for the other approaches can all be reduced by a factor of L, the degree of parallelism. For the HS approach, if the packet forwarding delays are dominant compared with packet processing times, the search time will be approximately proportional to $N\bar{t}_p(N/L)/L$. For the WM approach, the search time in (20) is proportional to N/L, but very large L might cause a processing bottleneck at the NMS which controls the search.

The search time for the SM approach is also proportional to N/L and can be reduced by increasing the number of mobile agents, L. It is not clear from (24) that higher L will necessarily increase the traffic burden on NEs; although each NE will be visited by a mobile agent more often, the average size of mobile agents will be shorter (they do not accumulate as much state because the search time is shorter).

VI. Conclusion

Although results are not surprising, this study attempted to provide a quantitative approach to evaluating different network management approaches. The traditional CS approach appeared to be worse than the distributed approaches in performance and scalability. The HS approach with a sufficient number of midlevel managers can successfully eliminate the traffic and processing bottleneck of the CS approach, but the benefits have to be weighed against the practical costs of deploying more midlevel managers. The routine monitoring application did not stress the advantages of the WM approach because the capability to dynamically delegate code was unnecessary. For data searching, the WM approach can perform well by exploiting parallelism, but the NMS can be a potential bottleneck. The SM approach has the potential to perform well for monitoring but not if the additional overhead required for strong security is substantial. Similarly for data searching, the potential benefits of mobile agents have to weighed against the additional costs related to security and migration of code and state.

The study supports the argument that a single network management approach will not be best for all types of applications. Instead, the most appropriate approach for a specific application should be selected after a careful evaluation of the costs following the modeling framework here.

References

[1] J. Case, M. Fedor, M. Schoffstall, and J. Davin, "A Simple Network Management Protocol (SNMP)," IETF, RFC 1157, 1990.

[2] "Information Technology—Open Systems Interconnection-Common Management Information Protocol Definition," ITU-T, Geneva, ITU Recommendation X.711, 1992.

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

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

[5] M. Baldi, S. Gai, and G. Picco, "Exploiting code mobility in decentralized and flexible network management," presented at the Int. Workshop on Mobile Agents, Berlin, Apr. 1997.

[6] B. Joy *et al.*, *The Java Language Specification*, 2nd ed. Reading, MA: Addison-Wesley, 2000.

[7] S. Vinoski, "CORBA: Integrating diverse applications within distributed heterogeneous environments," *IEEE Commun. Mag.*, vol. 35, pp. 46–55, Feb. 1997.

[8] M. Greenberg, J. Byington, and D. Harper, "Mobile agents and security," *IEEE Commun. Mag.*, vol. 36, pp. 76–85, July 1998.

[9] V. Pham and A. Karmouch, "Mobile software agents: An overview," *IEEE Commun. Mag.*, vol. 36, pp. 26–37, July 1998.

[10] D. Milojicic, F. Douglis, and R. Wheeler, *Mobility Processes, Computers and Agents*, D. Milojicic, F. Douglis, and R. Wheeler, Eds: ACM Press, 1999.

[11] D. Milojicic *et al.*, "MASIF: The OMG mobile agent system interoperability facility," in *Proc. 2nd Int. Workshop on Mobile Agents*, Sept. 1998, pp. 50–67.

[12] A. Fuggetta, G. Picco, and G. Vigna, "Understanding code mobility," *IEEE Trans. Software Eng.*, vol. 24, pp. 342–361, May 1998.

[13] A. Carzaniga, G. Picco, and G. Vigna, "Designing distributed applications with mobile code paradigms," in *Int. Conf. Software Engineering*, 1997, pp. 22–32.

[14] M. Baldi and G. Picco, "Evaluating the tradeoffs of mobile code design paradigms in network management applications," in *Proc. ICSE'98*, Kyoto, Japan, Apr. 19–25, 1998, pp. 146–155.

[15] A. Liotta, G. Knight, and G. Pavlou, "Modeling network and system monitoring over the internet with mobile agents," in *Proc. NOMS'98*, 1998, pp. 303–312.

[16] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser, "Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)," IETF, RFC 1906, 1996.

[17] M. Siegl and G. Trausmuth, "Hierarchical network management: A concept and its prototype in SNMPv2," *Comput. Netw. and ISDN Syst.*, vol. 28, pp. 441–452, Apr. 1996.

[18] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser, "Coexistence Between Version 1 and Version 2 of the Internet-Standard Network Management Framework," IETF, RFC 1908, 1996.

[19] K. Meyer *et al.*, "Decentralizing control and intelligence in network management," in *Proc. 4th Int. Symp. Integrated Network Management*, May 1995, pp. 4–16.

[20] J. Stamos and D. Gifford, "Remote evaluation," *ACM Trans. Prog. Lang. and Syst.*, vol. 12, pp. 537–565, Oct. 1990.

[21] A. Bieszczad, B. Pagurek, and T. White, "Mobile agents for network management," *IEEE Commun. Surveys*, vol. 1, no. 4Q, pp. 2–9, 1998.

[22] U. Blumenthal and B. Wijnen, "User-Based Security Model (USM) for Version 3 of the Simple Network Management Protocol (SNMPv3)," IETF, RFC 2574, 1999.

[23] G. McGraw and E. Felten, *Java Security: Hostile Applets, Holes, and Antidotes*. New York: Wiley, 1997.

[24] L. Gong *et al.*, "Going beyond the sandbox: An overview of the new security architecture in the Java development kit 1.2," presented at the USENIX Symp. Internet Technology and Systems, Dec. 1997.
[25] C. Lai *et al.*, "User authentication and authorization in the java platform," presented at the 15th Annual Computer Security Applications Conf., Dec. 1999.
[26] M. Powell and B. Miller, "Process migration in DEMOS/MP," in *Proc. 9th ACM Symp. Oper. System Principles*, Oct. 1983, pp. 110–119.

Thomas M. Chen (S'81–M'88–SM'96) received the B.S. and M.S. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge, MA, and the Ph.D. degree in electrical engineering from the University of California, Berkeley

He is an Associate Professor in the Department of Electrical Engineering and a faculty affiliate of the Linda and Mitch Hart e-Center at Southern Methodist University, Dallas, TX. Prior to joining SMU, he worked on ATM research at GTE Laboratories (now Verizon), Waltham, MA. He is a senior technical editor for *IEEE Network*, a senior technical editor for *IEEE Communications Magazine*, past founding editor of *IEEE Communications Surveys* and an associate editor for *ACM Transactions on Internet Technology*. He is the coauthor of the monograph, *ATM Switching Systems* (Artech House, Norwood, MA, 1995).

Dr. Chen was the recipient of the IEEE Communication Society's Fred W. Ellersick best paper award, in 1996.

Stephen S. Liu (S'74–M'79–SM'88) received the B.S. and Ph.D. degrees in electrical engineering from National Cheng-Kung University, Taiwan, and Georgia Institute of Technology, Atlanta, GA, respectively.

He codeveloped the ISO and ANSI standard CRC-32 error detection code polynomial used on Ethernet and various high-speed data networks and coauthored the book, *ATM Switching Systems* (Artech House, Norwood, MA, 1995). Currently, he is a member of Technical Staff in the NGN Backbone Group, Verizon Technology Organization. His current interests are in optical networking technologies.