Active Distributed Monitoring for Dynamic Large-scale Networks

Antonio Liotta°, George Pavlou
Center for Communication Systems Research
University of Surrey, Guildford, GU2 7XH, UK
{A.Liotta, G.Pavlou}@eim.surrey.ac.uk

Graham Knight
Computer Science Department
University College London, London WC1 6BT, UK
G.Knight@cs.ucl.ac.uk

Abstract – Networks offering services of high availability and quality need to be carefully monitored. Their increasing size and complexity stresses the ability of currently used static centralized systems. Decentralized approaches are possible and a key issue is the placement of area monitoring stations for optimal operation. Previous research has resulted in computationally expensive algorithms that require a global centralized network view. In this paper we propose a much simpler distributed algorithm and show that it performs as well as existing near-optimal but expensive, centralized algorithms. In addition, we propose that area monitoring stations are mobile agents, cloning and optimally placing themselves by executing the proposed algorithm. As network conditions change, e.g. through faults or persisting congestion, agents can adapt and migrate to new locations. We quantify the benefits of our approach against both the centralized and centrally-computed static distributed approaches.

I. INTRODUCTION

The ability to monitor a network or distributed system accurately and effectively is of paramount importance for its operation, maintenance and control. Network monitoring entails the collection of traffic information used for a variety of performance management activities e.g. capacity planning and traffic flow predictions, bottleneck and congestion identification, quality of service monitoring for services based on service level agreements, etc. A key aspect is that collection of traffic information should be supported in a timely manner, so that reaction to performance problems is possible, and without incurring too much additional traffic on the managed network. Given these constraints, efficient network monitoring is an interesting research problem. In this article we are highlighting the limitations of existing solutions and we propose an approach that uses the emerging paradigm of mobile software agents.

The conventional approach to network monitoring is to poll managed network devices from a *centralized* management station where most of the intelligence is concentrated. An event-driven approach, in which managed devices "alarm" the management station, is also possible but requires complex functionality to be built into network elements. This increases their complexity and cost and, more importantly, it is fixed and cannot be customized and augmented as requirements change. As such, the polling model is being widely used because of its simplicity and flexibility. Polling-based centralized monitoring inherits the intrinsic limitations of centralized systems, namely reduced responsiveness, accuracy and scalability.

An alternative approach is to divide the network into smaller "areas" and deploy one polling-based monitoring station per area, with all of those stations reporting in an event-based manner to a centralized station. This results in a two-level monitoring hierarchy, with the centralized station at the second level of the hierarchy having a network-wide view. We term this the *static decentralized* approach as the locations of the area monitoring stations are typically computed off-line and do not change after deployment. This scales better but still results in reduced flexibility, in particular for dynamic, large scale network topologies. The ideal solution is dynamic or *active distributed* monitoring, with stations computing their optimal location based on the target monitored objects. A station will move to that location and will possibly adapt to network changes and move again when conditions change in order to maintain optimality. Given the required mobility, proactivity and reactivity properties, a monitoring station could be realized through a mobile software agent.

A key issue in the static decentralized approach is the calculation of the optimal locations for the monitoring stations. Optimality in this case concerns the minimization of network traffic incurred due to (localized) polling and the minimization of the required latency in collecting the necessary information. The problem regarding the optimal placement of a number of servers in a large network has been studied in the literature. It is equivalent to p-median and p-center problems in graph theory. These problems are both NP-complete when striving for optimality. Approximate algorithms from graph theory exist but they are characterized by polynomial complexity of high degree. An additional problem is that these algorithms are centralized, requiring the network distance matrix at the monitoring station. While this is less of a problem in offline calculations for medium to long-term optimal locations, it becomes an important problem for active distributed

° This work was partly undertaken in the context of the IST MANTRIP project, which is partially funded by the Commission of the European Union.

solutions in which optimal locations need to be (re-)calculated by the agents themselves.

Given our proposal to use mobile software agents as area monitoring stations, a distributed algorithm is required. In this paper we propose such an algorithm that relies on agents learning about the network topology through node routing table information which is accessed through standard management interfaces. The monitoring system is deployed through a "clone and send" process starting at the centralized network-wide station. The same algorithm adopted for the initial agent deployment is also used for agents to adapt to network changes through migration. Key features of this algorithm are its distributed nature, i.e. each agent carries and runs the algorithm, and its low computational complexity.

We evaluate our algorithm through simulation and we compare it to the centralized and also to the centrally-computed near-optimal static distributed approaches. We quantify the relative benefits of our approach and show the superior scalability in comparison to the centralized approach and the almost equal properties in comparison to the near-optimal static distributed approach, which is much more expensive computationally to calculate. We also show how agents can adapt to faults, re-calculate their optimal location and migrate there, maintaining optimality.

II. The Problem of Dynamically Computing the Location of the Area Managers

A major requirement on our active distributed monitoring system is its ability to place the area managers efficiently. This imposes the following two constraints: 1) the agents implementing the monitoring operation have to be placed in strategic locations which result in near-minimal overall incurred traffic and/or response time; 2) the agent deployment time must be characterized by a predictable upper-bound which must, in turn, be smaller than the overall duration of the monitoring operation itself.

The first problem is analogous to the problem of locating multiple emergency facilities in a transport network. In that case, emergency facilities need to be placed in a way that minimizes either the total traveling cost (p-median problem) or the maximum traveling time (p-center problem). In the case of distributed monitoring, the total traveling cost is the equivalent of the total *traffic* incurred by the agents to perform the monitoring operation. For instance, if agents follow the "polling" technique, the cost associated to *request* and *response* packets will be proportional to the packet size and to the total number of links traversed by those packets.

Similarly, maximum traveling time is the equivalent of the *response time* of the monitoring system. In the case of polling-based monitoring, this is the span of time elapsed between issuing a request packet by the agent and the arrival of the corresponding response packet.

Both the p-median and the p-center problems have been extensively studied since the seventies (see [1] for a comprehensive review). Those problems have been proved NP-hard on a general network, though a number of approximate solutions characterized by polynomial complexity have been proposed so far.

Algorithms based on heuristics tend to be more computationally efficient, though they do not guarantee near-optimality and may even be far from optimality. Four relatively efficient algorithms that solve the p-median problem are described in [2]: The *myopic* algorithm, despite not being necessarily optimal, is appealing for its simplicity. It requires $O(p*N^2)$ operations to locate p agents in a network having N nodes. The *neighborhood search* algorithm is computed in at least $O(N^2)$ operations. The *exchange* algorithm requires at least $O(p*N*(N-p))$ operations. Finally, an optimization-based *lagrangian* algorithm is reportedly more complex but is still useful because it often provides results that are either provably optimal or near-optimal. Furthermore, Daskin presents also an algorithm that finds an approximate solution to the p-center problem in $O(N^2*p^{1/2})$ time.

Nevertheless, the location algorithms found in the literature are not suitable to solve the agent location problem for at least three reasons. Firstly, they are computationally too complex to suit the requirements of large-scale networks. Secondly, they do not provide guaranteed upper-bounds on computational time. This makes it difficult, if not impossible, to judge whether the active distributed monitoring approach is feasible for relatively short monitoring operations. Finally, and even more importantly, those algorithms are computed on the network distance matrix. Hence, they can only be computed at a single node that must retain an up-to-date version of such information. Naturally, that node would be the one where the monitoring station is located. But this requirement is obviously unrealistic for large-scale networks.

III. An Approach to Effective Placement of Area Managers

This section presents a novel approach to the location problem, which addresses the shortcomings of existing location algorithms. In the proposed solution, the location of area managers is neither fixed nor pre-determined at design time. Area managers are realized with mobile agents, simple autonomous software entities that, having access to network routing information, can adapt and roam through the network. The distributed monitoring system is deployed by progressively partitioning the network and by populating each partition with monitoring agents.

We assume the existence of an agent system supporting *strong mobility* – i.e., the ability of agents to preserve their state during migration – and *agent cloning* – i.e. the ability of agents to create and dispatch copies, or 'clones', of themselves. Agents are assumed to have access to routing information obtainable from network routers through standard network management interfaces. We also assume that MA hosts - i.e., locations in which MAs are able to run - are evenly distributed within the network. This, in other terms, means that for each router there is always an MA host that is located relatively close to it and, for each LAN, the number of MA hosts is proportional to the number of MOs that need to be monitored in that LAN. Under these assumptions, the MA distribution tree —i.e., the set of routes used for MA deployment— does not differ significantly from the routing tree rooted at the monitoring station. Without loss of generality, we envisage a scenario in which routers can act as MA hosts during MA deployment. In such case, the MA distribution tree would actually coincide with the routing tree.

```
1 Get monitoring task specification (at the
    monitoring station)
2 Generate 1 MA implementing the task
    * set MA parameters to the values extracted
        from the task specification
3 Extract list of MOs from current MA
4 For each MO extract routing information from
    local router
    * get next-hop node id from current MA
        location to MO
    * get cost to reach MO from current MA
        location
5 Estimate cost for the current MA to monitor
    its associated MOs
6 Use the estimated cost to compute the number of
    MAs to be cloned by current MA and clone them
7 Decompose task of current MA into a number of
    sub-tasks equal to the new number of current
    MAs
8 For each current MA
    * set its task to one of the above sub-tasks
    * set its list of MOs to a disjoint subset of
        the total current MOs
    * estimate cost to start monitoring from
        current location
    * estimate cost to monitor from one neighbor
        location
    * IF (lowest cost is from current location)
        THEN start MA
    * ELSE {
              migrate to the cheapest location
              GOTO 3
          }
```

Figure 1. Proposed agent location algorithm.

The agent location process is described in Figure 1. The *number* and *location* of mobile agents is computed by subsequently comparing the monitoring task parameters with routing information extracted from network routers. Starting from the monitoring station, the list of monitored objects (MOs) is matched against *next-hop* addresses and *routing costs* to reach those MOs from the current location. This simple matching operation is sufficient for the agent to create a first partitioning of the network. A number of agents equal to the number of partitions are cloned. Each agent is assigned to a different partition and is configured to monitor the subset of the MOs belonging to that partition. Then, each agent autonomously resumes the "partitioning & cloning" process that ends when the number of MOs per partition falls below a given heuristic threshold. The algorithm can be further illustrated by showing the basic steps performed in the case of the simple network depicted in Figure 2. Those steps are depicted in Figure 3.

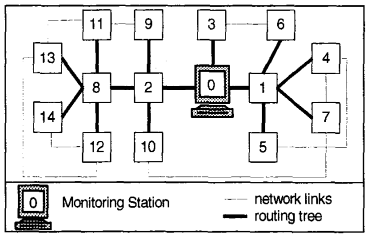

Figure 2. Sample network topology.

Initially, the manager delegates a given task to one agent and starts it at the monitoring station (a). By extracting routing information from the local router and matching them with the list of MOs, this agent estimates the need for an extra agent. An agent is thus cloned, and the original task is decomposed into two sub-tasks, including the redistribution of the MOs between the two agents (b). Then each agent autonomously searches its the best location and migrates to it (c). The agent in location 1 is now ready to start since it has estimated that its current location is the one with minimum cost. In contrast, the agent in location 2 decides to share its task with another agent and clones it (d). The decomposition/migration process starts again leading to one agent running in node 2 and the other migrating to node 8 (e). Eventually, the agent in location 8 has found its cheapest location and start executing (f).

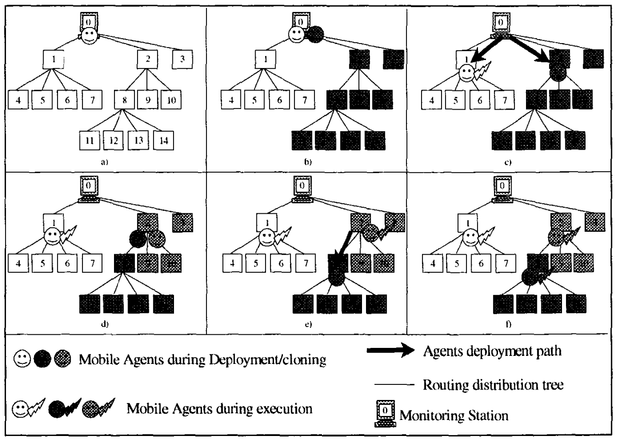
a)
b)
c)
d)
e)
f)
Mobile Agents during Deployment/cloning
Agents deployment path
Routing distribution tree
Mobile Agents during execution
Monitoring Station

Figure 3. Example agent deployment process.

It should be noted that the use of cloning results in minimal traffic around the monitoring station. In fact, only two agents leave the station, although the resulting number of agents is three. The cloning algorithm is executed in a distributed fashion (on nodes 1, 2, and 8). Finally, the processing is performed in parallel among nodes at the same level (1 and 2). This algorithm is computed dynamically in the sense that the final agent location depends critically on the network status detected at deployment time. In our implementation, agents keep sensing the network during the whole duration of their life and can adapt to changes in network topology through migration.

IV. EVALUATION OF ACTIVE DISTRIBUTED MONITORING

A. Method of Evaluation

Four different aspects of the proposed active monitoring system are evaluated. First, the asymptotic complexity of the agent location algorithm is analytically assessed.

Then, the distance from optimality of the agent location algorithm is evaluated by simulation. The reference near-optimal agent location is computed using the software package SITATION [2], configured to run the *lagrangian* location algorithm, which guarantees at least near-optimality. We also generated the agent locations randomly to emulate the worse possible agent distribution.

Realistic network topologies, composed of routers, links, and hosts have been generated using the GT-ITM topology generator [3-5]. In particular, transit-stub topologies resembling the Internet topology and having 16, 25, 32, 50, 64, 75, and 100 nodes respectively, have been generated in order to assess the sensitivity of the location algorithm to network size.

In order to simulate IP network and protocol behavior we have adopted the NS-2 simulator from U.C. Berkeley/LBNL [6] and extended it with Mobile Agent capabilities. Agent migration and cloning have been implemented along with the actual agent location algorithm, which is incorporated in each agent. This algorithm has been optimized to minimize the total incurred monitoring traffic. Total hop-distance and maximum weighted distances have been measured for increasing "agents to number of monitored objects" ratios. Those metrics are directly related to the total traffic incurred by the monitoring system and to its response time.

An important feature of distributed monitoring systems is their ability to scale better than their centralized counterparts. To quantify the potential benefits we have measured traffic and response time for increasing values of polling rate, number of monitored nodes, network diameter, and number of agents. Due to lack of space though we report only the first case.

Finally, we have carried out a simple experiment to assess the adaptability of the system to network failures. We have simulated a scenario in which two links located in the proximity of the central monitoring station go down. We measured traffic and response time at steady state – i.e., after both after both the centralized and the distributed system had adapted to the new network topology. In each of the above simulations, experiments were repeated a sufficient number of times to ensure the validity of the results, which were also statistically validated.

B. Area Managers Configuration Time

The agent location is actually computed during agent deployment. Hence, the algorithmic asymptotic complexity can be estimated by looking at the predominant factors involved from start up until all agents are deployed.

Steps 1-2 of Figure 1 are performed at the monitoring station, at start up time. Their predominant factor is the cloning time, $CLON_{time}$. In contrast, steps 3-8 may be repeated at subsequent levels of the routing distribution tree (rooted at the monitoring station). They will be repeated at most $R(u)$ times, whereby $R(u)$ is the network radius. Agents running at the same level of the distribution tree, execute independently from each other, in separate physical locations. Hence the computational complexity of the location algorithm can be determined by considering the part that is inherently sequential. Therefore, the complexity is $R(u)$ times the complexity of steps 3-8.

Upon arriving at a node, an agent needs to be de-serialized and instantiated, before executing from step 3. This operation takes a constant time, $DESERIL_{time}$. Steps 3-4 require a number of iterations equal, at most, to the total number of monitored nodes. The dominant cost for each iteration is given by the look-up operation to the routing table to extract the *next_hop* and the *cost* values. Thus, the total contribution of steps 3-4 is $c*O(N)$, where c accounts for one look-up time. Step 5 involves a number of iterations which, in the worst case, is equal to the maximum node degree, δ_{max} that in typical networks is significantly smaller than the number of nodes and, typically, does not increase with N. The iterations of steps 6-8 are actually performed as part of steps 5 and in the worst case involve the process of cloning and configuring δ_{max} new agents. Cloning will take a constant time, $CLON_{time}$; the reassignment of the monitored nodes takes a constant time too because it reuses information initially processed during Steps 3-4. Finally, each new agent will require a serialization time, $SERIAL_{time}$ before being sent to its destination. The latter will add a forwarding delay, $FORW_{time}$ and a transmission time, $TRANSM_{time}$.

Therefore the agent deployment time, $DEPL_{time}$ that actually coincides with the time to compute the agent location algorithm, can be expressed as:

$DEPL_{time} = \{DESERIL_{time} + c*O(N) + \delta_{max}* [CLON_{time} + SERIAL_{time}] + TRANSM_{time} + FORW_{time}\}*O(R(u)) = c_1 *O(N * R(u)) + c_2 * O(R(u)) \propto O(N*R(u))$.

In practice, c_1 is of the order of at most 10E-6 seconds, since the current technology allows for a number of look-up operations of at least 10E6 per second. c_2 is in the order of seconds since with current mobile agent platforms $[TRANSM_{time} + FORW_{time}]$ is typically in the order of 10E-3 to 10E-1 seconds and $[DESERIL_{time}+ CLON_{time}+ SERIAL_{time}]$ is in the order of seconds or fraction of seconds [9]. Therefore, if $N << 10E6$ then $[c*O(N)] << \{DESERIL_{time} + \delta_{max}*[CLON_{time} + SERIAL_{time}] + TRANSM_{time} + FORW_{time}\}$ and, consequently, $DEPL_{time} \approx c_2 * O(R(u))$. In this case the deployment term will predominate over the computational one and $DEPL_{time}$ will be in the order of seconds times $O(R(u))$.

The results of the above analysis prove that the algorithm is $O(N*R(u))$ in general while it is $O(R(u))$ in practice. Since $R(u)$ is typically sub-linear with N, the proposed algorithm is sub-linear with *N*. The algorithm asymptotic complexity does not vary with the number of MAs *p* since each agent computes its location in parallel and independently from the others. This is conditional to the assumption that the system where an agent executes has sufficient memory and computational resources to meet the agent's requirements. These results represent a significant improvement over the centralized location algorithms reported in the literature.

C. Comparison with Other Server Location Algorithms

The distance from optimality of the proposed location algorithm can be evaluated by observing the plots of Figure 4. The total hop-distance is directly related to the total steady-state monitoring traffic. It can be observed that the proposed location algorithm leads to traffic values that are always smaller than those that would be achieved with the lagrangian algorithm, which is provably near-optimal. Hence, our agent-based algorithm is near-optimal too. In particular, a percentage improvement in the range of 0-3% was measured. It should be stressed once again that the lagrangian algorithm cannot be used to solve the agent location algorithm for the reasons already mentioned.

It should be noted that, for the sake of completeness, we simulated situations characterized by up to a large number of agents (p/N=0.4). However for a more efficient resource utilization, typical "agents to nodes" ratios are envisioned to be much smaller (p/N<0.1). The fact that the total hop-distance achieved by placing the agents in a random fashion is very far from our near-optimal solution (38-48% difference for p/N<0.1) provides another good justification for the adoption of the agent-based approach. The percentage reduction in traffic with respect to centralized polling (p/N=0) is also significant. For instance, for p/N=0.1 the reduction in traffic will be greater than 30% and will increase monotonically with p/N.

Finally, the fact that the three curves tend to converge for large values of p/N is not unexpected since when p/N=1 the number of agents equals the number of nodes. Hence, each of the three location algorithm will equally succeed in placing the agent evenly. The plot which reports the maximum weighted distance (directly related to response time) for the three location algorithms is qualitatively analogous to the previous one. However, in this case the agent location curve, though very close to the near-optimal one, does not exhibit any inferior value. In particular, the distance from near-optimality is 0-5% for p/N<0.1. This result was expected since the simulated agent location algorithm was optimized to minimize traffic, not response time. Further simulations, not reported here for brevity, proved that near-optimality with respect to response time can be achieved with trivial modifications to the agent algorithm.

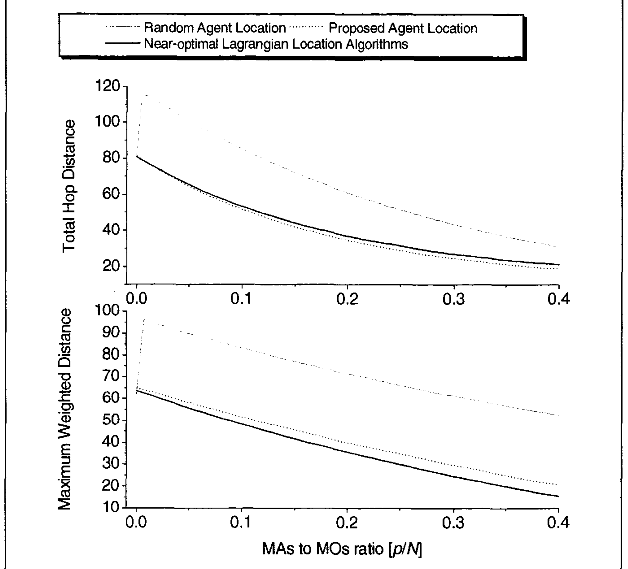

Figure 4. Distance from near-optimality.

D. Scalability with respect to Centralized Monitoring

This section evaluates the scalability of the proposed monitoring system from a different viewpoint than the one of Section B. We previously assessed how well the agent deployment algorithm scaled to draw conclusions on its viability. Herein, we evaluate scalability at steady-state by comparing the agent monitoring system with a conventional centralized system. Expectedly, an improvement in performance is achieved with the former approach due to its intrinsic distributed nature. However, the results of our simulations provide a quantitative evaluation.

Figure 5 shows traffic and response time measurements achieved with centralized and distributed polling-based monitoring, respectively. Increasing values of polling rate are required for larger accuracy and timeliness, but incur increasing volume of traffic. In our scenario p/N=0.1, whilst network diameter and average node degree are kept constant, hence the linear behavior. It can be observed that the agent solution leads to an approximate 50% reduction in traffic and 28% reduction in response time. Another aspect of scalability is the maximum polling rate that can be sustained by the network. Our simulations showed that the agent solution could sustain polling rates of the order of 200% larger than its centralized counterpart. We then assessed the sensitivity to p/N, keeping all the other parameter unchanged. This time the curves, not shown for brevity, exhibited a non linear behavior. Both traffic and response time decreased significantly for agent configuration having 0<p/N<0.15. However, for larger values of p/N the improvement was negligible. We concluded that a larger portion of agents is neither convenient nor useful. In fact, the larger is the number of agents, the larger the agent deployment overheads.

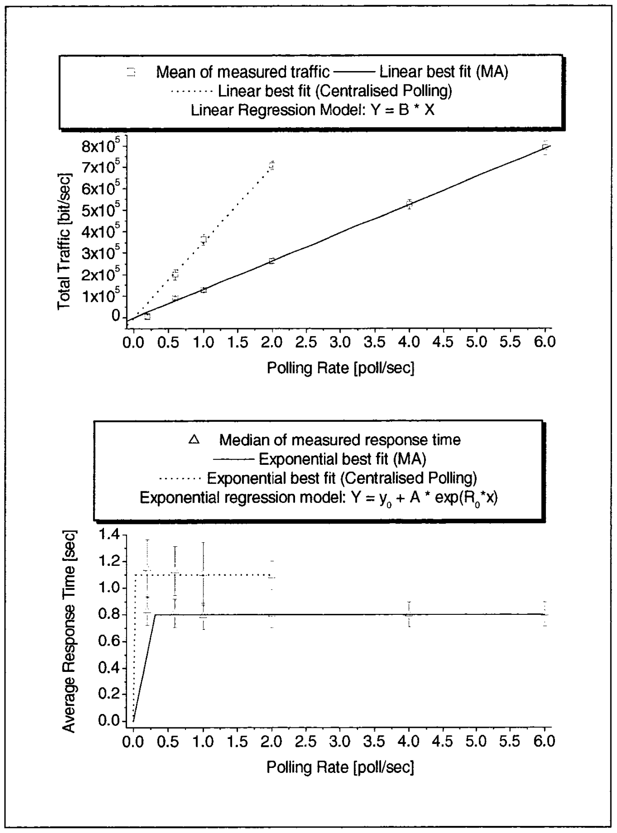

Figure 5. Scalability.

E. Adaptability to Network Dynamics

The ability of a monitoring system to adapt to network changes is a very attractive property, especially in view of the dynamic behavior of current and future networks. Network congestion and failures, along with mobile computing result in rapidly changing network topologies.

The conventional approach is to achieve adaptability by dynamically changing the routing tree rooted at the monitoring station. This is performed by the routing protocols. Consequently, as a result of congestion or failures, monitoring packets get re-routed through generally longer paths and both traffic and response times tend to deteriorate.

In active monitoring, agents keep sensing the network during their operation and can periodically estimate the cost of alternative locations. Agent migration is triggered when the cost reduction justifies the migration overheads.

In our implementation, agents adopt the same logic used during deployment time to sense the network and estimate costs associated to candidate neighbor nodes. This adaptation strategy is based solely on local decision. Locality has the advantage of simplicity but has the drawback of not considering more global optimization strategies.

We have simulated a simple scenario in which 2 links located in the vicinity of the central monitoring station fail. Traffic and response time were measured before the failure. After the failure the routing protocol readjusted the routing tables and full connectivity was achieved. In addition, the agent system reconfigured itself by relocating some of the agents. Steady-state traffic and response time were measured again.

Figure 6 shows the snapshot of those performance indicators taken before and after the link failure, respectively. With the centralized polling solution, both request and response packets get re-routed through longer paths. Consequently, both traffic and response time increase significantly – they almost doubled in our scenario. On the contrary, with the agent system the performance degradation at steady state is in the order of 5-10%.

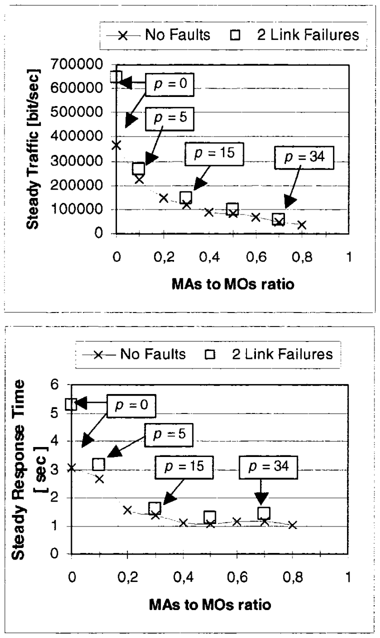

Figure 6. Adaptability.

It should be mentioned, though that our experiments on adaptation are only at their early stage and will require more extensive simulation before generalized conclusions can be drawn. In particular agent migration, if not properly triggered, may lead to instability that can be disastrous. Furthermore, migration policies including more sophisticated heuristics need to be studied.

V. DISCUSSION AND CONCLUSIONS

As networks become all pervasive, the importance of efficient monitoring can only increase. "Network-centric" approaches, such as Sun's Jini architecture [7] envisage large numbers of comparatively simple devices (cell-phones, televisions, thermostats) all accessible across the net. Management systems in the future will need to keep track of these devices and determine which are present, which are functioning correctly and so on. Of additional relevance to this article is the fact that Jini devices can host mobile code.

However, the introduction of code mobility represents a serious paradigm shift which has not yet found widespread acceptance in the management arena. The most common objection to code mobility has been the lack of guarantees with regard to security and safety. We have not considered these issues. Instead, we have tried to assess the advantages, in terms of reduced traffic and shorter response times that code mobility might bring.

The feasibility and the advantages of shifting to the 'active decentralized' monitoring paradigm can be seen from our simulation-based analysis of an example system based on autonomous mobile agents. In this system, the decision as to where best to locate the agents makes use of 'local' information in a 'distributed' fashion. Each agent identifies a better location and decides whether to 'clone' further agents or not - the decision is based solely upon local routing information. Each agent acts autonomously to solve a portion of the global configuration problem, which is solved in a distributed fashion.

Our study has shown that a system based on mobile agents will out-perform a conventional system in many typical situations. It has also helped in identifying conditions in which agents cannot be recommended. Mobile agents were particularly effective in reducing the communication and processing bottleneck located at the monitoring station. Despite the simplicity of the agent configuration algorithm, agents end-up evenly distributed for network topologies resembling internetworks and hierarchical networks, respectively. Consequently, the achieved improved scalability and performance are not entirely surprising. However, our analysis provides a quantitative comparison of performance and scalability between centralized monitoring and agent-based distributed monitoring. The main factor limiting the use of agents is their migration overhead - clearly, migration traffic and migration time are not incurred in the centralized approach. The former depends on agent size which, in turn, depends on the complexity of agent functionality. Simple autonomous agents able to perform local aggregation of monitored data, generate statistics, and trigger alarms can be implemented in the Java language in modules of size in the order of 10 Kbytes [8].

Agent migration time is dominated by the agent serialization/de-serialization process, required to prepare an agent for transmission and execution, respectively. Depending on various implementation aspects, this process requires times in the order of hundreds of milliseconds, well above the tens of milliseconds required to transport the code over the wire [8, 9]. For this reason, the time to complete the deployment of all the agents will be of the order of seconds or tens of seconds, depending on the number of agents, and on the network size and average node degree. Traffic bursts incurred during the initial deployment or long deployment times may not be acceptable in some situations. For instance, monitoring tasks whose total duration is of the order of agent deployment time cannot be implemented using this approach.

Our analysis highlights a key advantage of using 'full' code mobility instead of degenerate cases or stationary agents i.e. adaptability. We have presented an example showing how agent migration can be used to implement a monitoring system adaptable to network changes, a particularly attractive property for dynamic networks. Again, the agent migration overheads identified in our analysis give a clue as to the time-scales over which adaptation might be effective. Since agent migration time is in the order of a second, agents are suitable to compensate to changes within time-scales larger than a second.

In conclusion, the application of code mobility to network monitoring is feasible, and generally results in increased scalability and steady-state performance. However, there are conditions in which agents cannot compete with the simplicity and efficiency of the conventional centralized approach – e.g. in very short or very simple monitoring operations. Therefore, the ideal solution is to integrate code mobility into existing systems and benefit from the relevant advantages rather than rely completely on mobile software agent approaches. A seamless integration between existing and mobile agent based approaches should be the topic of future research.

While our approach has been devised for network monitoring, it can also be applied to a whole set of similar problems related to the optimal location of p servers in a network of N nodes, with information exchanges required between them and $p << N$.

REFERENCES

[1] B. C. Tansel, R. L. Francis, T. J. Lowe, "Location on Networks: a Survey", *Management Science*, Vol.29(4), (April, 1983), pp.482-511.

[2] M. S. Daskin, "Network and Discrete Location", *Wiley*, (1995).

[3] E.W. Zegura, K.L. Calvert, M.J. Donahoo, "A Quantitative Comparison of Graph-based Models for Internet Topology", *IEEE/ACM Transactions on Networking*, (1997).

[4] E.W. Zegura, K.L. Calvert, S. Bhattacharjee, "How to Model an Internetwork", *IEEE INFOCOM 96*, San Francisco, CA, USA, (1996).

[5] K.L. Calvert, M.B. Doar, E.W. Zegura, "Modeling Internet Topology", *IEEE Communications Magazine*, (June, 1997).

[6] K. Fall, K. Varadhan, "NS Notes and Documents", UC Berkeley, (http:// www.isi.edu/ ~salehi/ ns_doc/) (October 1999).

[7] J. Waldo, "The Jini Architecture for Network-centric Computing", *Communications of the ACM*, Vol. 42(7), pp. 76-82, (July, 1999).

[8] G. Knight, R. Hazemi, "Mobile Agent based management in the INSERT project", *Journal of Network and System Management* (Mobile Agent-based Network and Service Management), Vol. 7 (3), (September, 1999).

[9] C. Bohoris. A. Liotta, G. Pavlou, "Evaluation of Constrained Mobility for Programmability in Network Management", Proc. of *DSOM 2000*, pp. 243-257 (December, 2000)