

# Distributed Network Monitoring For Evolving IP Networks

Marina Thottan, Li (Erran) Li, Bin Yao, Vahab S. Mirrokni and Sanjoy Paul

**Abstract**—Evolving monitoring infrastructure in response to network growth is a critical aspect of network management. Previous work in network management primarily focused on optimizing monitoring systems for static networks. In this paper, we address the problem of optimally upgrading the existing monitoring infrastructure as the network evolves. The problem formulation presented here captures the trade off between adding new monitoring resources vs. disrupting the existing infrastructure. We show that this problem is NP hard and not approximable within a factor better than  $n^\epsilon$  in the general case and no better than  $\log(n)$  when only shortest paths are considered. We develop a heuristic algorithm and evaluate its performance using simulated network evolution scenarios. We show that in spite of not allowing poller relocation, our adaptive on-line algorithm has comparable performance to that of the offline algorithm where such constraint does not exist.

## I. INTRODUCTION

The capability of monitoring a network allows network operators to obtain up-to-date status information of the network, and be promptly notified when network changes occur. The continuous availability of such information is important to many aspects of network operation, for example, failure and bottleneck detection, service level agreement (SLA) verification, quality of service (QoS) guarantees, etc [1]. On the other hand, as networks are becoming larger and more complex, so are the monitoring systems. Therefore, a scalable monitoring infrastructure that can continuously monitor the network is of great interest to network operators. Real network also evolves, making it necessary to adjust the monitoring system such that the network can be continuously monitored. In such cases, from the perspective of network operators, it is desirable to adapt the monitoring system at a minimum cost.

Commercial network management systems are primarily based on SNMP (Simple Network Management Protocol) [10]. SNMP has two kinds of entities: a management station, and management agents running on network nodes. The management station sends SNMP commands to management agents to obtain information about the network. Traditionally, the management station function is performed by a centralized poller responsible for polling all network nodes. In large networks, such an architecture can easily result in overload of the poller and of links close to the poller [1].

To improve the scalability of SNMP, several enhancements have been made to the SNMP protocol itself by improving

efficiency of basic SNMP primitives [4]. In addition, both distributed and hierarchical polling architectures [15], [16], [9] have been proposed. One example of the distributed approach is management by delegation, where management tasks are delegated to individual nodes. To further improve scalability, most commercial systems today use a three-level hierarchical polling architecture [11], as shown in Figure(1). The lowest level (pollers) perform the actual task of polling. Each poller is responsible for a polling domain which consists of multiple network nodes. Information collected by the poller is summarized and then forwarded to the aggregators. The aggregator further synthesizes information from multiple pollers and forwards it to the central management station [7]. Within this architecture one can obtain algorithmic solutions

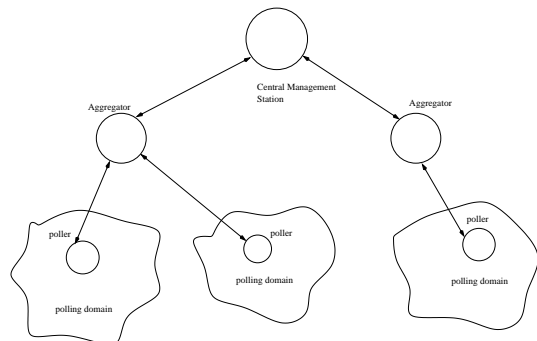


Fig. 1. Hierarchical Management Architecture

that specifically address the scalability problems. In particular, our previous work [6] addresses the problem of minimizing the number of polling stations while keeping the polling bandwidth within predefined limits. By minimizing the number of pollers, we limit the cost of the monitoring infrastructure. Furthermore, the bandwidth constraint on the links prevents hot spots near the pollers. However, in this and in other previous work [3], [5] only fixed network topologies are considered. In this paper, we analyze the cost associated with adapting an existing monitoring system to changes in the network. We develop a cost function that incorporates tradeoff between adding new pollers to the system versus re-configuring the existing pollers [9]. The re-configuration cost on existing pollers arises from the cost of tracking pollees assigned to each poller. That is, when a poller to pollee assignment is changed, various components of the hierarchical management system need to be updated. For example, when the polling domain of a node changes it is necessary to update the aggregator databases since they use the summarized

Marina Thottan, Li (Erran) Li, Bin Yao, and Sanjoy Paul are with the Center for Networking Research, Bell Labs, Holmdel NJ 07733. Email: {marinat, erranli, byao, sanjoy}@dnrc.bell-labs.com

Vahab S. Mirrokni is with the Laboratory for Computer Science at MIT. Email: mirrokni@theory.lcs.mit.edu

information regarding several polling domains from multiple pollers to compute end-to-end performance metrics [11][13].

In this paper, we formulate the problem of finding the minimum cost of adjusting the monitoring system in response to incremental network changes. We present an adaptive algorithm that explicitly takes into consideration the cost trade-off issues, and evaluate its performance using evolving network topologies. The performance of our adaptive algorithm is compared with the offline version[6]. Even in the offline case, we prove that the problem is NP hard and cannot be approximated to within a factor better than  $n^\epsilon$  in the general case, and no better than  $\log(n)$  when only shortest paths are considered.

The paper is organized as follows: related work is discussed in Section(II) where we provide an overview of existing approaches to scalable network management systems. The problem formulation and complexity analysis are presented in Section(III). In Section(IV) we present our heuristic algorithm that addresses the problem of monitoring growing networks in a cost-effective manner. The performance of our heuristic algorithm is evaluated through simulations and results are presented and discussed in Section(V).

## II. RELATED WORK

Existing algorithmic work in this area attempt to address the issue of hot spots in the network due to bandwidth bottlenecks. Several algorithms have been studied which focus on reducing polling traffic on the network. For example, it is possible to identify a minimum number of nodes that must be monitored to measure the average bandwidth utilization in the network [3]. Also, it is possible to identify the minimum number of probes that must be used to obtain latency measurements on the network [3]. By reducing the number of nodes to be polled and the total number of probes used it is possible to considerably reduce measurement traffic. The problem of optimal placement of measurement instrumentation is addressed in [5]. In this work, using knowledge of a fixed network topology, optimal location for network instrumentation is obtained using the constraint of a maximal center to node distance. The authors in [5] also provide an algorithm for identifying the location of the instrumentation given a fixed number of measurement centers.

All of the above approaches do not address the issue of bandwidth constraint explicitly and therefore cannot guarantee against bandwidth bottlenecks. This problem is addressed by the Bandwidth Constrained Poller Placement algorithm presented in [6]. In this work, given a network topology, it is possible to identify a minimum set of pollers that are subject to a constraint on the maximum polling bandwidth per link. This approach helps reduce the cost of the network infrastructure and the bandwidth constraint ensures that there will be no network hot spots generated as a result of the polling traffic. In [6] the problem of poller placement as wells as poller-pollee assignment scheme was evaluated only for a static network. However, with the increasing pace of network growth, there is a need to evolve the existing monitoring infrastructure to meet the demands of the new network topology. We propose an

algorithm that addresses this problem by explicitly modeling the costs involved in upgrading the management system.

## III. PROBLEM FORMULATION

### A. Network Model

We use two undirected graphs,  $G_1 = (V_1, E_1)$  and  $G_2 = (V_2, E_2)$ , to represent the original network and the evolved network, respectively.  $V_1$  and  $V_2$  are the set of nodes (or routers) that must be polled in  $G_1$  and  $G_2$ , and  $E_1$  and  $E_2$  are the set of links. In this work we assume that SNMP managers (pollers) can be co-located with network nodes. By co-located we mean that the polling stations are located on one of the interfaces of the routers.  $o_{ij}$  and  $x_{ij}$  is the poller-pollee assignment in the original graph  $G_1$  and the new network  $G_2$  respectively.  $o_{ij} = 1$  if and only if  $j$  is polled by  $i$  in  $G_1$  otherwise  $o_{ij} = 0$ . Similarly,  $x_{ij} = 1$  if and only if  $j$  is polled by  $i$  in  $G_2$  and  $x_{ij} = 0$  otherwise.  $b(e_{ij})$  and  $c(e_{ij})$  represent the bandwidth constraint and capacity for each link. Notice that the bandwidth constraint is typically a fraction of the total capacity on the link.  $l(v_i)$  represents the bandwidth required to poll each node.  $l(v_i)$  depends on the number of interfaces being monitored at node  $v_i$  and may be different for each node. The polling load is a function of the number of interfaces on the router, the number of paths in the network and the number of customers and services that are supported by the node.

### B. Cost Function

When the network changes from  $G_1$  to  $G_2$ , the path between a poller and pollee pair may change. Some pollees may be deleted from the network and new pollees added. Consequently, a poller may not be able to poll its pollees because there is not enough bandwidth on the new path to reach the pollee. Furthermore, newly added nodes should be assigned a poller as well. As a result, to be able to poll every node in  $V_2$ , some existing pollers may need to be reconfigured, or new pollers may need to be introduced.

Adding a new poller is clearly a cost to the network operator, it includes the cost of purchasing additional hardware and configuring the poller. Reconfiguring an existing poller can be a cost as well [9]. Hierarchical polling implementations require that in order to maintain a single point of control in the management system, the aggregator databases must be aware of all the poller-pollee assignments within their aggregation domain. This is essential to calculate end-to-end performance metrics. Therefore the event of a change in the poller-pollee assignment requires administrator intervention and could result in loss of network information. Therefore, we assume that the amount of changes in the pollee set does not affect the cost associated with reconfiguring a poller. In other words, if  $p$  has 100 pollees, we assume that changing 1 or 99 of these pollees incurs relatively the same cost (viz), the cost of reconfiguring a single existing poller  $p$ .

Clearly, the cost of adding a new poller ( $C_a$ ) and the cost of reconfiguring an existing poller ( $C_r$ ) are different. We use a parameter  $\gamma = C_r/C_a$  to capture the ratio between these two. The total cost of adjusting the polling infrastructure in

response to network change is the sum of the cost for newly added pollers and that for reconfiguring existing pollers. Since cost is an expense to a network operator, it is desirable to minimize it. Note that in the event that a poller in  $G_1$  is removed in  $G_2$ , the pollees associated with the removed poller will fall into the group of unassigned pollees.

More formally, we define the cost function as:

$$C(\gamma, G_1, G_2) = \sum_{\forall v_j \in V_2} (\gamma_j - a_j) + \gamma \sum_{v_i \in V_1} I(\sum_{v_j \in V_1} |(o_{ij} - x_{ij})|) \quad (1)$$

where,  $I$  is the indicator function;  $\gamma_j = 1$  if node  $v_j$  is a poller in  $V_2$ , and 0 otherwise.  $a_j = 1$  if node  $v_j$  is a poller in  $V_1$ , and 0 otherwise.  $o_{ij} = 1$  if node  $v_j$  is polled by node  $v_i$  in  $V_1$ , and 0 otherwise.  $x_{ij} = 1$  if node  $v_j$  is polled by node  $v_i$  in  $V_2$ , and 0 otherwise.

### C. Integer Programming Formulation

The optimal poller selection and pollee assignment problem can therefore be stated as follows: given a new network  $G_2$ , the original poller set  $Y$  and the original poller-pollee assignment scheme  $O$ , determine a poller-pollee assignment scheme  $X$  in  $G_2$  such that:

- the cost of changing from  $O$  to  $X$  is minimum;
- all nodes in  $G_2$  are polled;
- the allocated bandwidth on each link should be less than its bandwidth constraint, i.e.,  $b(e_{ij})$ ;

The problem formulation presented above can be cast into an integer programming problem with the assumption that the routes between the poller and the pollee are fixed. This assumption is reasonable since in a typical service provider network the routing paths are fairly stable. Our objective is:

$$\text{Minimize } C(\gamma, G_1, G_2) \quad (2)$$

subject to the following constraints:

$$\sum_{j=1}^n x_{ij} = 1, \text{ for each } v_i \in V \quad (3)$$

$$x_{ij} \leq y_j, \text{ for each } v_i, v_j \in V \quad (4)$$

$$\sum_i^n \sum_j^n \delta_e^{ij} w_i x_{ij} \leq b(e) \text{ for each } e \in E \quad (5)$$

$$x_{ij} \in \{0, 1\}, \text{ for each } v_i, v_j \in V \quad (6)$$

$$y_j \in \{0, 1\}, \text{ for each } v_j \in V \quad (7)$$

Where  $n = |V_2|$  is the total number of nodes in the network; The binary variable  $x_{ij}$ , indicates whether node  $v_i$  is polled by node  $v_j$ , where  $v_i, v_j \in V_2$ . The binary variable  $\delta_e^{ij}$  indicates whether edge  $e$  belongs to the path  $P_{ij}$  between node  $v_i$  and  $v_j$ .  $P_{ij}$  represents the path between the pollee node  $i$  and the poller node  $j$ .  $w_i$  represents the polling bandwidth required to poll node  $v_i$  and  $b_e$  corresponds to the bandwidth constraint on the edge (physical link)  $e$ .

The first constraint makes sure that each node  $v_i$  is assigned to exactly one poller. The second constraint guarantees that a node  $v_j$  must be a poller if some other node  $v_i$  is assigned to it. The third constraint ensures that the sum of the polling bandwidth used by all the poller pollee pairs on each link does not exceed its allocation.

### D. Complexity Issues

In this section, we provide the complexity analysis of the poller placement problem. The online algorithm discussed in this paper is a special case of the offline poller placement problem (minPL) where  $G_1$  is empty. Li et. al. [6] show that minPL is NP-hard, thus we do not hope to get an exact polynomial time algorithm for minPL. One can only hope to get an approximation algorithm with a guaranteed approximation factor. An  $\alpha$ -approximation algorithm is an algorithm whose solution is at most  $\alpha$  times the cost of the optimum solution.

First we restate the minPL problem from [6]. Given a graph  $G = (V, E)$ , a path  $P_{ij}$  for any pair of vertices  $i$  and  $j$ , a bandwidth constraint or capacity  $b(e_{ij})$  for every edge  $e$ , and a demand  $d_i = l(v_i)$  i.e. the bandwidth requirement<sup>1</sup> on each vertex, the minPL problem is to find the minimum number of pollers such that all demands on other vertices can be satisfied using the given paths while respecting the bandwidth constraints.

We generally consider the special case in which given paths are necessarily shortest paths. We call this special case S-minPL problem. We state the following strong inapproximability results for minPL and S-minPL problems. Details of the proofs can be found in [12].

*Theorem 3.1:* The minPL problem is not approximable within a factor better than  $n^\epsilon$  for some  $\epsilon < 1$ .

*Theorem 3.2:* The S-minPL is not approximable within a factor better than  $\log(n)$ .

## IV. A COST EFFECTIVE GREEDY HEURISTIC ALGORITHM

Section III shows that the problem of obtaining optimal poller adjustment strategy under bandwidth constraint in evolving networks is NP-hard. More precisely, given the old network  $G_1$  and the new network  $G_2$ , optimizing the cost function  $C(\gamma, G_1, G_2)$ , as defined in Section III, is NP hard and cannot be approximated better than a factor  $\log(n)$  of the optimal. We therefore examine polynomial time heuristics in this section. Note that the objective of this heuristic is to minimize the cost function, which includes both poller cost and reconfiguration cost. This is different from our previous work that deals only with static network topologies, where the number of pollers is minimized.

It is clear from Definition 1 that the cost function is dependent on the value of  $\gamma$ , which captures the relative cost of adding a new poller vs. reconfiguring an existing poller.  $\gamma$  is the ratio of the two costs and thus provides the flexibility to tune the cost function for any given network scenario. The size and complexity of each network is different, resulting in different cost of adding and reconfiguring pollers. Therefore, the value of  $\gamma$  can be different for each network operator.  $\gamma$  may also change over time, as hardware price fluctuates and the capability of network management software evolves. We therefore designed our heuristic in such a way that it can work well with a wide range of  $\gamma$ .

<sup>1</sup>In this section, we use vertex as the pollee and demand as the bandwidth requirement on vertices

The heuristic presented in this section tries to minimize  $C$ , the cost function, while making sure that all pollees, new and old, are assigned to a poller. The heuristic operates in two steps: The first step, *pre-processing*, tries to obtain a large set of pollers that do not need to be reconfigured, *i.e.*, they can continue to poll their original pollee set in the evolved network. Intuitively, this reduces the poller reconfiguration cost. The second step, *Poller selection for unassigned pollees*, uses a greedy heuristic to select pollers for the remaining pollees in a cost efficient way.

#### A. Pre-processing

The preprocessing step takes as input  $G_1$ ,  $G_2$ , and the original poller-pollee mapping  $o_{ij}$ , where  $o_{ij} = 1$  iff node  $j$  is polled by node  $i$  in  $G_1$ . It produces a set of unassigned pollees  $U$  and a set of pollers  $D$  that need to be reconfigured. Here, we mention the important aspects of this algorithm.

The algorithm first identifies existing pollers that can no longer poll its complete set of pollees. This could happen because a pollee may be removed in  $G_2$  or the path between a pollee and its poller changed in  $G_2$  and there is not enough bandwidth on the new path. Such pollers are called “disturbed pollers” and are put in set  $D$ . In the case where a poller  $v_p$  is removed as the network evolves, that is  $v_p \in G_1, v_p \notin G_2$ , we can simply treat  $v_p$ ’s original pollees in  $G_2$  as newly introduced pollees. Pollers in  $G_1$  that are not disturbed are called “un-disturbed pollers”. The set of pollees originally assigned to pollers in  $D$  that are also in  $G_2$ , and the newly introduced nodes in  $G_2$ , make up the set  $U$ , the unassigned pollee set.

Each step of this algorithm is highlighted here and details are shown in Algo IV-A. First, we initialize  $U$  to the set of new pollees and  $D$  to be empty. At step 2, we make as unassigned those pollees in  $G_2$  whose old pollers in  $G_1$  are not present in  $G_2$ . At step 3, we find all pollers whose set of pollees or their paths are different in  $G_2$ . For such a poller  $v_p$ , we compute  $U_p$  *i.e.*, the set of pollees assigned to it and add  $v_p$  to the set of disturbed pollers and add all their assigned pollees to the set of unassigned pollees. At step 4, for each poller not in  $D$ , assign its pollees to itself. At step 5, we try to restore the original poller-pollee relations for pollees whose network path have changed. Only when such assignments cannot be achieved due to bandwidth constraints, does the poller become a disturbed poller.

#### B. Greedy Poller Selection for Unassigned Pollees

This algorithm takes  $G_1, G_2, U, D, o_{ij}$  as input and produce  $x_{ij}$ , the new poller-pollee assignment in  $G_2$ . The objective is to find a poller-pollee assignment such that the increase in cost function  $C(\gamma, G_1, G_2)$  is minimized. Our greedy heuristic iteratively assigns pollees in  $U$  to a poller such that the average increase in the cost function per added pollee is the smallest. For example, if poller  $v_{p_1}$  can poll 3 additional pollees with average cost being 0.5, and poller  $v_{p_2}$  can poll 2 additional pollees and average cost per pollee is 0.4, the heuristic would pick node  $v_{p_2}$  as the new poller.

#### Algo V-A: Algorithm for Pre-processing

Input:  $G_1, G_2, o_{ij}$   
Output:  $U, D$

1.  $U = \{V_2 - V_1\}, D = \emptyset$
2. for each poller  $v_p \in V_1$ , and  $v_p \notin V_2$ ,  
 $U_p = \{v_i | v_i \in V_1 \cap V_2 \text{ and } o_{pi} = 1\}$   
 $U = U \cup U_p$
3. identify all pollers  $v_p \in V_1$ , s.t.  
 $H_p \neq \emptyset$ , where  $H_p = \{v_i | v_i \in V_1, o_{pi} = 1, \text{ and either } v_p \notin V_2 \text{ or the path between } v_i \text{ and } v_p \text{ in } G_1 \text{ is different from that in } G_2\}$   
 $U_p = \{v_i | v_i \in V_1 \cap V_2 \text{ and } o_{pi} = 1\}$   
 $D = D \cup \{v_p\}, U = U \cup U_p$
4.  $\forall v_p \notin D$  and  $v_p$  is a poller, assign its original pollees and adjust available bandwidth on each link
5. for each  $v_p \in D$ , if  $\forall v_i$  s.t.  $o_{pi} = 1, v_i \in V_2$  and  $U_p$  can be polled by  $v_p$  in  $G_2$ ,  
 $D = D - \{v_p\}, U = U - U_p$ , and adjust available bandwidth on each link

The details of this greedy algorithm can be found in Algo IV-B. The first step performs initializations based on the input. At step 2, the heuristic first use an algorithm, *ChoosePollee* to calculates for every node  $v_i$  in  $G_2$ , the set  $A_i$  of pollees in  $U$  that  $v_i$  can poll. Then a cost function is calculated for each  $v_i$ . Every node  $v_i$  in  $G_2$  belongs to one of four categories: disturbed poller, undisturbed pollers, unassigned pollee, and undisturbed pollees that are assigned to undisturbed pollers. If  $v_i$  is a disturbed poller, using  $v_i$  to poll additional nodes does not bring additional cost. When  $v_i$  is a undisturbed poller, the cost is  $\gamma$ , which is the cost of reconfiguring a poller. If  $v_i$  is an unassigned pollee, the cost of making  $v_i$  a poller is 1. Finally, if  $v_i$  is the pollee of an un-disturbed poller, making  $v_i$  a poller requires reconfiguring  $v_i$ ’s poller and adding a poller, hence the cost is  $1 + \gamma$ . At step 2.2, when  $v_i$  is selected, it is added to the disturbed poller set  $D$ , as reconfiguring it further does not bring additional cost.  $v_i$ ’s new pollees are removed from  $U$  and then, after each  $v_i$  is selected, pollees polled by pollers in  $D$  are re-arranged among  $D$  using the shuffle presented in [6]. The heuristic is repeated if there are still unassigned pollees.

#### Algo V-B: Greedy Poller Selection for Unassigned Pollees

Input:  $G_1, G_2, o_{ij}, U, D$   
Output:  $x_{ij}$

1. for  $v_i \notin D, x_{ij} = o_{ij}$  for all  $v_j$   
for  $v_i \in D, x_{ij} = 0$  for all  $v_j$   
 $U_0 = U$
2. while  $U \neq \emptyset$  do
  - 2.1 for every  $v_i \in V_2$ , use *ChoosePollee* to calculate the set  $A_i$  of additional pollee in  $U$  that  $v_i$  can poll in  $G_2$   
Assign a cost function  $C_i$  for  $v_i$ :  
if  $v_i \in U_0, c(i) = 1$   
if  $v_i \in D, c(i) = 0$   
if  $\exists v_j \neq v_i$  s.t.  $o_{ij} = 1$  and  $v_i \notin D, c(i) = \gamma$   
if  $\exists v_j \neq v_i$  and  $v_i \notin D$  and  $v_j \notin D$  s.t.  $o_{ji} = 1, c(i) = 1 + \gamma$   
from  $\{v_i | A_i \neq \emptyset\}$ , select  $v_i$  s.t.  $c_i / |A_i|$  is minimum
  - 2.2 make  $v_i$  a poller in  $G_2$  if it is not already a poller and assign  $A_i$  to  $v_i$ :  
 $D = D \cup \{v_i\}, U = U - A_i$
  - 2.3 Apply *ShuffleAndReduce* procedure to pollees polled by  $D$

Note that at each iteration, there is at least one pollee in  $U$  that gets assigned to a poller. Therefore, there are at most  $U$  steps and each step has polynomial complexity. It follows that this heuristic is guaranteed to finish in polynomial time. It is greedy because it tries to minimize the average cost of newly

added pollees at each step, and it only considers the case of reconfiguring or adding a single poller.

### C. Heuristics for ChoosePollee

S-MaxPollee[6] can be used as *ChoosePollee* at the second step of the algorithm. It calculates, for a given node  $v_p$ , the maximum number of pollees that can be assigned to  $v_p$  without violating bandwidth constraint. The other natural approach is for a fixed poller, find out the maximum amount of polling demand that can be assigned to it. We call this problem MaxDemand, and the special case restricted to shortest paths S-MaxDemand. We developed a 2-approximation algorithm for S-MaxDemand and used it as *ChoosePollee*. The performance of these two approaches are compared in Section V. We state the following theorems related to MaxDemand here and details of the proofs can be found in [12].

*Theorem 4.1:* MaxDemand is NP hard and is not approximable within a factor of  $n^\epsilon$  for some  $\epsilon > 0$ .

*Theorem 4.2:* S-MaxDemand is NP hard.

*Theorem 4.3:* The following greedy algorithm is a 2-approximation for S-MaxDemand problem:

- 1) Sort demands in decreasing order,  $d_1 \geq d_2 \geq \dots \geq d_n$ .
- 2) for  $i$  from 1 to  $n$ , satisfy  $d_i$  (assign  $d_i$  to the poller) if possible.

## V. SIMULATION RESULTS AND DISCUSSIONS

In this section, we evaluate the performance of the our adaptive heuristic algorithm on several different topologies and parameter settings. For simplicity, we make the reasonable assumption of shortest path routing. The focus of our work is to optimize the measurement infrastructure for evolving IP networks. The evolution of the network is captured by incrementing in steps the number of nodes in the network topology. We consider stepwise increments of 25 and 100 nodes. The polling load for each node is proportional to the number of interfaces on the node and the number of MPLS paths that originate from that node.

### A. Simulation Setup

In our simulations we assume a flat topology and all the links are assumed to have the capacity of OC-48. We generate topologies using the GT-ITM topology generator [17] and the BRITE topology generator [8]. The network graph used in this study is the Waxman model [14] and the Barabasi-Albert model [2]. We generate different network topologies by varying the Waxman parameter  $\beta$  for a fixed parameter  $\alpha$ . The varying  $\beta$  gives topologies with degrees of connectivity ranging from 4 to 8 for a given number of nodes in the network. Each link allocates a certain fraction of their bandwidth for polling traffic based on the capacity constraint imposed for the link. In our simulation, the fraction is the same for all links and we call this parameter LINKF, and the default value is 5%. Simulation results presented here are averaged over 5 different topologies. To account for additional polling load due to the existence of MPLS tunnels we assume, that each interface or MPLS tunnel requires a polling bandwidth

of 4Kbps. The number of MPLS tunnels per node is randomly chosen from a range of [1,500]. Our simulations investigate the tradeoff between re-optimizing the measurement infrastructure and keeping the current infrastructure intact. The performance of our heuristic is measured in terms (1) average number of pollers required, (2) percentage of pollers disturbed and, (3) average per-link polling bandwidth consumed in Mbps.

### B. Comparison of S-MaxDemand and S-MaxPollee

We first compare the S-MaxDemand heuristic with S-MaxPollee in the simple off-line case. The solution of the offline case is the initial state for our on-line algorithm. We use both the Waxman model and the BarabasiAlbert model. As seen in Table I and II, S-MaxDemand can reduce the number of pollers needed by as much as 20%. The reason is as follows. S-MaxPollee tends to connect several pollees with small polling demands to a poller such that some pollees with high demands are isolated and cannot connect to any other poller. Therefore a poller is forced to be added to the isolated part of the network containing the high demand pollee. The performance improvement of S-MaxDemand over S-MaxPollee depends on the variability of polling demands. Notice that when all the demands are the same, S-MaxDemand and S-MaxPollee are the same algorithm. The performance improvement of S-MaxDemand can be much higher than S-MaxPollee if the variability of demands are much higher than in our simulation setting. Also note that as the link fraction for polling bandwidth increases the performance of S-MaxDemand and S-MaxPollee are comparable. This is because the inefficiency of S-MaxPollee mentioned above is unlikely to happen.

	Link Fraction	Number of Nodes		
		500	600	700
S-MaxDemand	2%	23	28	34
	5%	6	5	6
S-MaxPollee	2%	28	31	39
	5%	7	6	7

TABLE I

PERFORMANCE COMPARISON OF S-MAXDEMAND AND S-MAXPOLLEE IN TERMS OF THE NUMBER OF POLLERS: WAXMAN MODEL

	Link Fraction	Number of Nodes		
		500	600	700
S-MaxDemand	2%	16	17	21
	5%	6	6	7
S-MaxPollee	2%	19	21	26
	5%	6	7	7

TABLE II

PERFORMANCE COMPARISON OF S-MAXDEMAND AND S-MAXPOLLEE IN TERMS OF THE NUMBER OF POLLERS: BARABASI-ALBERT MODEL

### C. Evaluation of the Adaptive Algorithm for Network Growth

Since our results from the previous section are qualitatively the same for both topology generators. We use Waxman model

for the rest of our simulation results. Since the performance of S-MaxDemand and S-MaxPollee are similar for LNKF=5%, we choose one of them in our simulation results presented in this section. In our simulation the network evolves in 5 stages from 500 nodes to 1000 nodes, each stage corresponds to an addition of 100 nodes and its associated links. We first show the performance of our adaptive algorithm as a function of the tradeoff parameter  $\gamma$ .  $\gamma$  permits us to effectively capture the costs involved in adding new pollers and in reconfiguring the existing pollers. We also compare our current adaptive algorithm with the performance of off-line greedy poller placement algorithm [6]. As expected, in general, the offline algorithm performs better than the online version. This is due to the fact that the goal of the online algorithm is to minimize the disturbance to the existing monitoring infra-structure.

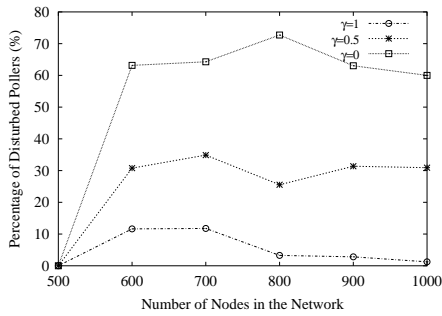


Fig. 2. Average Number of disturbed pollers in the system for varying  $\gamma$ .

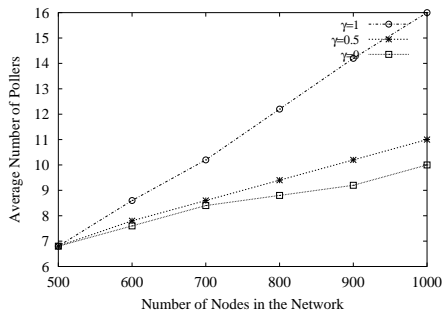


Fig. 3. Average Number of pollers in the system for varying  $\gamma$ .

1) *The Impact of  $\gamma$* : As shown in Figure 2, for a 1000 node network with  $\gamma = 1$  on the average we have, about 1% of the pollers disturbed. However, with  $\gamma = 0$  we have about 60% of the pollers disturbed. The tradeoff involved here is that as the number of disturbed pollers decreases, in order to meet the management requirements of the new topology we add more pollers to the overall system. Thus from Figure 3 the average number of pollers in the system for the 1000 node network with  $\gamma = 1$  is 16 while for  $\gamma = 0$  we have about 10 pollers.

We can see that as  $\gamma$  becomes larger, the number of pollers increases and the percentage of disturbed pollers decreases. Stated in another way, as the cost of reconfiguring a poller approaches that of adding a new poller, more new pollers are added to the monitoring system and less existing pollers are disturbed. This indicates that our heuristic successfully

captures the trade of between the two kind of costs. In practice, a network operator can calculate the value  $\gamma$  for its own network and plug it into the heuristic to obtain an adaptation strategy.

Figure 4 shows the average per-link polling bandwidth consumption for different values of  $\gamma$ . It is interesting to observe that despite having fewer number of pollers (about 10) in the system, the per-link polling bandwidth usage for  $\gamma = 0$  is smaller (up to 14%) than that for  $\gamma = 1$  (where the number of pollers is 16). This is due to the ShuffleAndReduce procedure used in Step(5) of the heuristic algorithm (see Algo IV-B). As the cost of disturbing the pollers is reduced, that is for lower values of  $\gamma$ , we have more and more pollees that can be shuffled to pollers that are near. Thus the average bandwidth consumption on the links decreases. The additional flexibility in the available polling bandwidth in turn reduces the total number of pollers required to meet the additional demands.

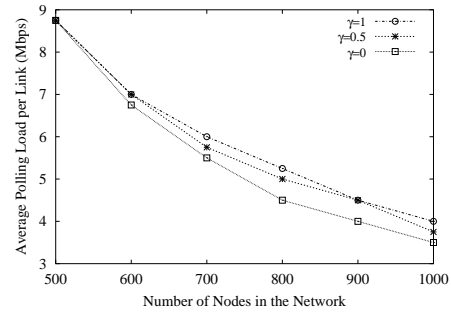


Fig. 4. Average polling bandwidth per link.

#### D. Comparison with the Off-line Greedy Poller Placement Algorithm

Figure 5 compares the performance of our adaptive algorithm with respect to the off-line poller placement scheme [6]. For comparison we choose to use  $\gamma = 0$  case, where we assume that there is no cost involved in disturbing the existing pollers. In the adaptive algorithm, this represents the worst case in terms of the total number of pollers required. As expected, the number of pollers in the off-line case is smaller than the number of pollers in the adaptive case. We emphasize this is due to the inefficiencies incurred by the cost constraint which prevents the relocation of existing pollers. However we find that the increase in the number of pollers is only 37% as compared with the off-line algorithm. Furthermore, the bandwidth consumption in the adaptive algorithm case is smaller (up to 16%) than the off-line case as illustrated in Figure 6. This shows that our adaptive algorithm is reasonably efficient.

#### E. Evolving Cost of Monitoring Infrastructure

Figure 7 shows the total cost for the polling system as a function of the stepwise increase in the number of network nodes. The non-linear increase in the cost function is due to the fact that as additional nodes are added to system(in steps

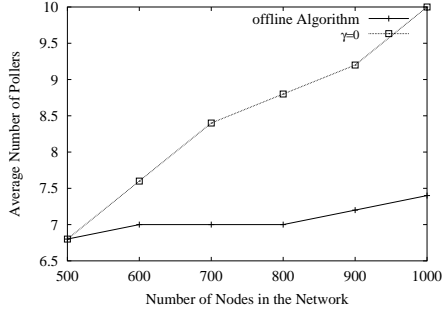


Fig. 5. Comparison with the off-line Algorithm: average Number of pollers in the system

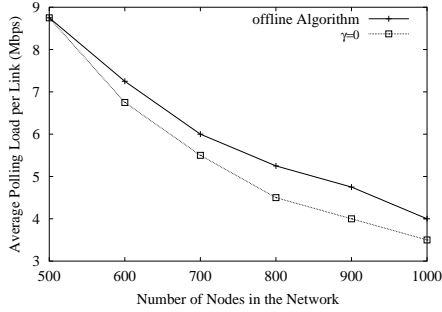


Fig. 6. Comparison with the off-line Algorithm: average polling bandwidth per link

of 25, 50, and 75), we also add more edges into the network thus increasing the bandwidth available for polling.

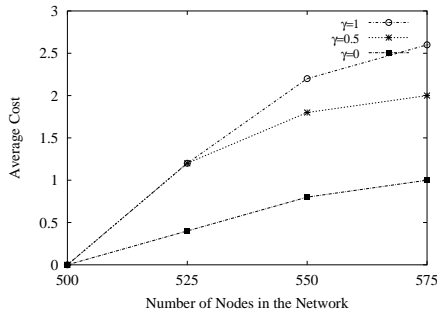


Fig. 7. Average cost as a function of stepwise increase in the number of nodes

### F. Sensitivity to Network Connectivity

We also observed in our simulation that as the network gets sparser, more pollers are required. In the case of a 1000 node network with the adaptive algorithm using  $\gamma = 0.5$ , for a fixed  $\alpha = 0.2$  and  $\beta$  being 0.08 and 0.15, the total number of pollers required was 15 and 11 respectively. We also observed that the difference in performance between the off-line and the adaptive case is insensitive to the degree of network connectivity.

### G. Application Scenarios

In some networks it is possible that there are some inherent hierarchical relationships such as the existence of routing domains (Autonomous Systems AS), and Point of Presence for the service provider (PoPs). These hierarchies may be natural places for poller placement. Such structures can be accounted for in our formulation. All we need is to set  $y_j = 1$  for each location  $u_j$ . We could also take into account fixed assignment between a poller  $u_j$  and a pollee  $u_i$  easily by setting  $x_{ij} = 1$ . Restrictions on assignment between  $u_j$  and  $u_i$  could be taken care of easily as well by setting  $x_{ij} = 0$ . These restriction can reflect the fact that it is not desirable for a poller to poll a pollee in another AS domain or OSPF area, etc. Our algorithm is able to account all these special cases with minor modification.

### VI. CONCLUSION AND FUTURE WORK

In this paper, we developed a function to measure the cost of evolving an existing network monitoring infrastructure. A parameter  $\gamma$  was introduced to capture the difference between the two kinds of costs: adding a new poller and reconfiguring an existing poller.  $\gamma$  is flexible and can be used to model any cost structure that is pertinent to the budgetary constraints imposed on the operations and maintenance of a network. We then formulated the problem of adapting the monitoring system as an integer programming problem. We obtained inapproximability results for this adaptation problem. We also developed a heuristic algorithm that takes  $\gamma$  into consideration. We evaluated our heuristic using realistic network models for varying values of  $\gamma$  and showed that it performed well, in terms of total number of pollers, against the off-line greedy poller placement algorithm. In spite of not allowing poller relocation and limiting disturbance to existing monitoring infrastructure, our adaptive on-line algorithm has comparable performance to that of the offline algorithm where such constraint does not exist.

We conjecture that there exists a  $\log(n)$  approximation for the off-line problem when all paths are shortest, and as part of our future efforts we would like to obtain such an algorithm. This approximation will provide a better benchmark to compare the performance of our adaptive scheme. We noticed that there exists a polynomial approximation scheme for the S-MaxDemand problem. Because the algorithm is involved, we plan to evaluate it as part of our future work. Additionally, we will work on the extension of our algorithm for fault tolerant polling, and to evaluate it for both link and poller failure resiliency. One possible approach is to use the routing protocol to precompute paths obtained by eliminating those edges that are prone to failures, and reserving bandwidth along these paths. We would also like to develop guidelines for determining appropriate values of  $\gamma$ , allowing our approach to be easily applied in practice.

### REFERENCES

- [1] A. Asgari, P. Trimintzios, M. Irons, G. Pavlou, and S. V. den Berghe R. Egan. A Scalable Real-Time Monitoring System for Supporting Traffic Engineering. In *Proc. IEEE Workshop on IP Operations and Management*, 2002.

- [2] A.L. Barabasi and R. Albert. Emergence of Scaling in Random Networks. *Science*, pages 509–512, 1999.
- [3] Y. Breitbart, C. Y. Chan, M. Garofalakis, R. Rastogi, and A. Silberschatz. Efficiently Monitoring Bandwidth and Latency in IP Networks. In *Proc. IEEE Infocom*, 2002.
- [4] D. Breitgand, D. Raz, and Y. Shavitt. SNMP GetPrev: An Efficient Way to Access Data in Large MIB Tables. In *Proc. IEEE infocom*, 2002.
- [5] S. Jamin, C. Jin, Y. Jin, D. Raz and Y. Shavitt, and L. Zhang. On the Placement of Internet Instrumentation. In *Proc. IEEE Infocom*, 2000.
- [6] L. Li, M. Thottan, B. Yao, S. Paul. Distributed Network Monitoring with Bounded Link Utilization in IP Networks. In *Proc. of IEEE Infocom*, 2003.
- [7] Y. J. Lin and M. C. Chan. A Scalable Monitoring Approach Based on Aggregation and refinement. *IEEE JSAC*, 20(4):677–690, May, 2002.
- [8] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: An Approach to Universal Topology Generation. In *Proc. of International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS)*, 2001.
- [9] E. Putrycz and G. Bernard. Client Side Reconfiguration on Software Components for load Balancing. In *Proc. Intl. Conference on Distributed Computing Systems, Workshop*, 2001.
- [10] W. Stallings. *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*. Addison Wesley Longman, Inc, 3 edition, 1999.
- [11] Lucent Technologies. Vitalsuite, <http://www.vitalsuite.com>.
- [12] M. Thottan, B. Yao L. E. Li, V. S. Mirrokni, and S. Paul. Distributed Network Monitoring For Evolving IP Networks. *Bell Labs Technical Memorandum*, 2003.
- [13] M. Thottan, G. K. Swanson, M. Cantone, T. K. Ho, J. Ren, and S. Paul. SEQUIN: An SNMP-Based MPLS Network Monitoring System. *Bell Labs Technical Journal*, 8(1):95–112, 2003.
- [14] B.M. Waxman. Routing of multipoint connections. *IEEE Journal of Selected Areas in Communication*, 6(9):1617–1622, 1988.
- [15] Y. Yemini. The OSI Network Management Model. *IEEE Communications Magazine*, pages 20–29, May,1993.
- [16] Y. Yemini, G. Goldschmidt, and S. Yemini. Network Management by Delegation. In *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management*, 1991.
- [17] E. W. Zegura, K. Calvert, and S. Bhattacharjee. How to Model an Internetwork. In *Proc. IEEE Infocom*, volume 2, pages 594–602, 1996.