Efficiently Monitoring Bandwidth and Latency in IP Networks

Yuri Breitbart, Chee-Yong Chan, Minos Garofalakis, Rajeev Rastogi, Avi Silberschatz

Abstract—**Effective monitoring of network utilization and performance indicators is a key enabling technology for proactive and reactive resource management, flexible accounting, and intelligent planning in next-generation IP networks. In this paper, we address the challenging problem of *efficiently* monitoring bandwidth utilization and path latencies in an IP data network. Unlike earlier approaches, our measurement architecture assumes a single point-of-control in the network (corresponding to the Network Operations Center) that is responsible for gathering bandwidth and latency information using widely-deployed management tools, like SNMP, RMON/NetFlow, and explicitly-routed IP probe packets. Our goal is to identify effective techniques for monitoring (a) bandwidth usage for a given set of links or packet flows, and (b) path latencies for a given set of paths, while minimizing the overhead imposed by the management tools on the underlying production network. We demonstrate that minimizing overheads under our measurement model gives rise to new combinatorial optimization problems, most of which prove to be $\mathcal{NP}$-hard. We also propose novel approximation algorithms for these optimization problems and prove guaranteed upper bounds on their worst-case performance. Our simulation results validate our approach, demonstrating the effectiveness of our novel monitoring algorithms over a wide range of network topologies.**

I. Introduction

THE explosive growth in Internet and intranet deployment for a constantly growing variety of applications has created a massive increase in demand for bandwidth, performance, predictable Quality of Service (QoS), and differentiated network services. Simultaneously, the need has emerged for measurement technology that will support this growth by providing IP network managers with effective tools for *monitoring* network utilization and performance. *Bandwidth* and *latency* are clearly the two key performance parameters and utilization indicators for any modern IP network. Knowledge of the up-to-date bandwidth utilizations and path latencies is critical for numerous important network management tasks, including application and user profiling, proactive and reactive resource management and traffic engineering, as well as providing and verifying QoS guarantees for end-user applications.

Indeed, these observations have led to a recent flurry of both research and industrial activity in the area of developing novel tools and infrastructures for measuring network bandwidth and latency parameters. Examples include SNMP and RMON measurement probes [1], Cisco's NetFlow tools [2], the IDMaps [3], [4] and Network Distance Maps [5] efforts for measuring end-to-end network latencies, the `pathchar` tool for estimating Internet link characteristics [6], [7], and packet-pair algorithms for measuring link bandwidth [8], [9]. A crucial requirement for such monitoring tools is that they be deployed in an intelligent manner in order to avoid placing undue strain on the shared resources of the production network.

As an example, Cisco's NetFlow measurement tool allows NetFlow-enabled routers to collect detailed traffic data on packet flows between source-destination node pairs [2]. NetFlow-enabled routers can generate large volumes of export data due to the size and distributed nature of large data networks, the granularity of the recorded flow data, and the rapid data traffic growth. The key mechanism for enhancing NetFlow data volume manageability is the careful planning of NetFlow deployment. Cisco suggests that NetFlow be deployed incrementally (i.e., interface by interface) and strategically (i.e., on carefully-chosen routers), instead of being widely deployed on every router in the network [2]. Cisco domain experts can work with customers to determine such "key" routers and interfaces for NetFlow deployment based on the customers' traffic flow patterns and network topology and architecture [2]. Similar observations hold for the deployment of SNMP agents [1], since processing SNMP queries can adversely impact router performance and SNMP data transfers can result in significant volumes of additional network traffic. In particular, as modern Network Management Systems (NMS) shift their focus toward service- and application-level management, the network monitoring process requires more data to be collected and at much higher frequencies. In such scenarios, the SNMP-polling frequency needs to be high enough not to miss relevant changes or degradations in application behavior or service availability [10]. (In fact, even for failure monitoring, Stallings [1] suggests that short polling intervals are often required in order for the NMS to be responsive to problems in the network.) When such high SNMP-polling frequencies are prescribed, the overhead that a polled SNMP agent imposes on the underlying router can be significant and can adversely impact the router's throughput. Further, the problem is only exacerbated for mid- to low-end routers (e.g., that implement large parts of their routing functionality in software). As an example, our experiments with a Cisco 4000-series router on our local network showed the throughput of the router to drop as much as $15-20\%$ during a polling cycle (where repeated `getnext` queries are issued to gather link utilization data). Obviously, polling such a router at reasonably high frequencies can severely impact its performance. Also, note that the *network bandwidth* consumed by such frequent SNMP polling for detailed router/application/service monitoring can be significant, primarily due to the large number of polling messages that need to traverse the network from/to the NMS to/from the polled routers. In fact, this is the main motivation behind work on distributed polling engines (e.g., [11]) and more recent proposals on "batching" SNMP-polling messages [10] and more effective SNMP-polling primitives [12].

As another motivating example, the IDMaps [3], [4] and Network Distance Maps [5] efforts aim to produce *"latency maps"* of the Internet by introducing measurement servers/tracers that continuously probe each other to determine their distance. In order to make their approach scale in terms of both the storage requirement and the extra probing load imposed on the network, both approaches suggest techniques for pruning the distance map based on heuristic observations [3], graph-theoretic ideas like t-spanners [4], or hierarchical clustering of the mea-

Information Sciences Research Center, Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974. {yuri, cychan, minos, rastogi, avi}@bell-labs.com

surement servers [5]. Minimizing monitoring overheads is also critical in order to avoid "Heisenberg effects", in which the additional traffic imposed by the network monitors perturbs the network's performance metrics and biases the resulting analysis in unpredictable ways [13].

In this paper, we address the challenging problem of *efficiently* monitoring bandwidth utilization and path latencies in an IP data network. Earlier proposals for measuring network utilization characteristics typically assume that the measurement instrumentation can be either (a) intelligently distributed at different points in the underlying network [3], [4], [5] or (b) placed at the endpoints (source and/or destination node) of the end-to-end path whose characteristics are of interest [6], [7], [8], [9]. In contrast, the monitoring algorithms proposed in this paper assume a much more common and, we believe, realistic measurement model in which a single, predefined point in the network (corresponding to the Network Operations Center (NOC)) is responsible for actively gathering bandwidth and latency information from network elements. Thus, rather than requiring the distribution of specialized instrumentation software and/or hardware (which can be cumbersome and expensive to deploy and manage) inside the production network, our algorithms enable a network administrator to efficiently monitor utilization statistics from a single point-of-control. More specifically, we propose effective, low-overhead strategies for collecting the following utilization statistics as a function of time:

1. *Bandwidth usage* for a given (sub)set of (a) *links*, and (b) aggregate *packet flows* between ingress-egress routers in the network. Link-bandwidth utilization information is obviously critical for a number of network management tasks, such as identifying and relieving congestion points in the network. Flow-bandwidth usage, on the other hand, provides bandwidth-utilization data at a finer granularity which can be invaluable, e.g., for usage-based customer billing and Service Level Agreement (SLA) verification.
2. *Path latencies* for a given (sub)set of (possibly overlapping) source-destination paths in the network. Once again, knowledge of the delays that packets experience along certain routes is important, e.g., in determining effective communication paths for applications with low-latency QoS requirements or dynamically routing the clients of a replicated service to their "closest" replica [3].

Our statistics collection methodology is based on exploiting existing, widely-deployed software tools for managing IP networks, like SNMP and RMON/NetFlow agents [1], [2] and explicitly-routed IP probe packets [14]. The target application domain for our monitoring strategies is large ISP networks, comprising hundreds of routers and several thousand network links. Such large ISP installations are typically characterized by high resource-utilization levels, which means that scalable monitoring strategies that minimize the impact of collecting utilization information on the underlying network are of the essence. This is especially true since this information needs to be collected periodically (e.g., every fifteen minutes) in order to continuously monitor the state and evolution of the network. The main contributions of our work can be summarized as follows.

• **Novel Algorithms for Efficiently Monitoring Link and Flow Bandwidth Utilization.** We demonstrate that the problem of collecting link-bandwidth utilization information from an underlying network while minimizing the required number of SNMP probes gives rise to a novel, $\mathcal{NP}$-hard generalization of the traditional Vertex Cover (VC) problem [15], termed *Weak VC*. Abstractly, Weak VC is a VC problem enriched with a linear system of equations for edge variables representing additional "knowledge" that can be used to reduce the size of the cover. We propose a new, polynomial-time heuristic algorithm for Weak VC that is *provably near-optimal* (with a logarithmic worst-case performance bound). Furthermore, we show that our heuristic is in fact very general and can be adapted to guarantee logarithmic approximation bounds for other $\mathcal{NP}$-hard problems that arise in efficient bandwidth monitoring, including the problem of minimizing the RMON/NetFlow overhead for collecting *flow-bandwidth usage information* from the network.

• **Novel Algorithms for Efficiently Monitoring Path Latencies.** We develop flexible techniques that are based on transmitting explicitly-routed IP probe packets from the NOC to accurately measure the latency of an arbitrary set of network paths. By allowing IP probes to be shared among the various paths, our probing techniques enable efficient measurement of their latencies. We prove that the problem of computing the (optimal) set of probes for measuring the latency of a set of paths that imposes minimum load on network links is $\mathcal{NP}$-hard. Fortunately, we are able to demonstrate that our optimal probe computation problem can be mapped to the well-known *Facility Location Problem (FLP)*, which allows us to use the polynomial-time approximation algorithm of Hochbaum [16] to obtain a *provably near-optimal* set of IP probes.

• **Simulation Results Validating our Monitoring Strategies.** In order to gauge the effectiveness of our monitoring algorithms, we have conducted a series of simulation experiments on a broad range of network graphs generated using the Waxman topology model [17]. For link-bandwidth measurements, we find that, compared to a naive approach based on simple VC, our Weak VC-based heuristic results in reductions as high as 57% in the number of SNMP-agent activations. Our experiences with latency measurements are similar, showing that, compared to naive probing strategies, our FLP-based heuristic returns sets of probes that, in several cases, traverse 20% fewer network links.

The remainder of this paper is organized as follows. Section II introduces our system model and the notational conventions used in the paper. The two optimization problems that we address in this paper are presented in Section III and Section IV, respectively, for the the link/flow bandwidth measurement problem and the path latency measurement problem. In Section V, we present simulation results to validate our proposed approach. Finally, we present our conclusions in Section VI. Due to space constraints, proofs of theoretical results can be found in the full version of this paper [18].

II. System Model and Notation

Our abstract model of a data network is an undirected graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ denotes the set of network nodes (i.e., routers) and $E = \{e_1, e_2, \ldots, e_m\}$ represents the set of edges (i.e., physical links) connecting the routers. We let $n = |V|$ and $m = |E|$ denote the number of G's nodes and edges, respectively, We also use $deg(v)$ to denote the *degree* (i.e., total number of incident edges) of node $v \in V$. The location of the Network Operations Center (NOC) is denoted by the "special" node v_0 where, without loss of generality, we assume that $v_0 \notin V$. Further, for a node $v_i \in V$, we denote the *shortest*

path (in terms of the number of links) from v_0 to v_i by β_i. Also, for paths p_i, p_j, $|p_i|$ is the number of links in p_i and $p_i \cdot p_j$ is the path resulting from the concatenation of p_i and p_j. Finally, given an edge e_i in G, $bw(e_i)$ stands for the bandwidth utilization at the corresponding link of the network. Table I summarizes the notation used throughout the paper with a brief description of its semantics. We provide detailed definitions of some of these parameters in the text. Additional notation will be introduced when necessary.

Symbol	Semantics
$G = (V, E)$	Network graph
$n = \|V\|, m = \|E\|$	Number of nodes/edges in G
$v_i, v_j, \ldots$	Generic nodes/routers in the network graph G
$e_i, e_j, \ldots$	Generic edges/links in the network graph G
$p_i, p_j, \ldots$	Generic paths in the network graph G
β_i	Shortest path from v_0 to v_i
$\|p_i\|$	Number of links in path p_i
$p_i \cdot p_j$	Concatenation of paths p_i and p_j
$\mathcal{F} = \{F_1, F_2, \ldots\}$	Set of traffic flows in the network
$\mathcal{F}_v$	Set of traffic flows routed through router v in G
$\mathcal{F}_e$	Set of traffic flows routed through link e in G
$bw(e)$	Bandwidth utilization for network link e
$bw(F)$	Bandwidth utilization for traffic flow F

TABLE I
NOTATION.

For our bandwidth-monitoring schemes that make use of flow information, we assume that all data traffic in the monitored network is distributed among a set of *packet flows* $\mathcal{F}$; that is, every data packet routed in G belongs to some flow $F_i \in \mathcal{F}$. Each such flow F_i is essentially a *directed path* from a source/ingress router to a destination/egress router in G. Note that, for a given pair of ingress-egress nodes, there may be multiple packet flows between them. Intuitively, each flow represents the aggregate traffic involving a set of source-destination IP address pairs. Edge ingress/egress routers typically serve a wide range of IP addresses and traffic between different source/destination addresses may be split at the network's edge routers along multiple flows, e.g., for traffic engineering or accounting purposes. We let $\mathcal{F}_v$ ($\mathcal{F}_e$) denote the set of packet flows routed through router v (resp., link e) in G. We also use $bw(F_i)$ to denote the bandwidth usage of flow $F_i \in \mathcal{F}$ in G.

III. Monitoring Link and Flow Bandwidth

An IP router is typically managed by activating an SNMP agent on that router. Over time, the SNMP agent collects various operational statistics for the router which are stored in the router's SNMP Management Information Base (MIB) and can be dispatched (on demand or periodically) to the NOC site [1]. SNMP MIBs store detailed information on the total number of bytes received or transmitted on each interface of an SNMP-enabled network element. Thus, periodically querying router SNMP MIBs provides us with a straightforward method of observing *link-bandwidth* usage in the underlying IP network. More specifically, assume that, using SNMP, we periodically download the total number of bytes received (b_{recv}) and bytes transmitted (b_{trans}) on a given router interface every t units of time. The average bandwidth usage for the incoming (outgoing) link attached to that interface over the measurement interval t is then b_{recv}/t (resp., b_{trans}/t). A naive, "brute-force" solution to our link-bandwidth monitoring problem would therefore consist of (1) activating an SNMP agent on every network router in G, and (2) periodically downloading the number of bytes observed on each interface to the NOC by issuing appropriate SNMP queries to all routers.

There are two serious problems with such a "brute-force" approach. First, running an SNMP agent and answering periodic SNMP queries from the NOC typically has a significant associated overhead that can adversely impact the performance characteristics of a router. Using such a naive bandwidth-monitoring strategy means that the routing performance of *every* router in the network is affected. Second, periodically downloading SNMP link-traffic data from every router can result in a substantial increase in the observed volume of network traffic. We are therefore interested in finding link-bandwidth monitoring schemes that minimize the SNMP overhead on the underlying IP network. More formally, our problem can be stated as follows.

Problem Statement [Low-Overhead Link-Bandwidth Monitoring]: Given a network $G = (V, E)$, determine a *minimum subset* of nodes $S \subseteq V$ such that enabling and monitoring SNMP agents on these nodes is sufficient to infer the link-bandwidth usage for every link of G[1]. ∎

For *flow-bandwidth* monitoring, RMON [1] or NetFlow agents [2] can be enabled on routers to measure the number of data packets shipped through any of the router's interfaces between specific pairs of source-destination IP addresses. Like SNMP, however, deploying and periodically querying RMON/NetFlow agents comes at a cost which can substantially impact the performance of the router and the observed volume of network traffic. In fact, both these problems are exacerbated for RMON and NetFlow compared to simple SNMP, since the measurements are collected and stored at a much finer granularity resulting in much larger volumes of management data. Thus, monitoring bandwidth usage at the level of packet flows gives rise to similar overhead-minimization problems.

In this section, we propose novel formulations and algorithmic solutions to the problem of low-overhead bandwidth monitoring for network links and packet flows.

A. A Vertex Cover Formulation

A simple examination of the naive method of activating SNMP agents on every network router reveals that it is really an overkill. Abstractly, to monitor all links in G, what is needed is to select a subset of SNMP-enabled routers such that every link in G is "covered"; that is, there is an SNMP agent running on at least one of the link's two endpoints. This is an instance of the well-known *Vertex Cover (VC)* problem over the network graph G [15]. Figure 1(a) depicts an example network graph and the nodes corresponding to a minimum VC of size 4. Even though VC is known to be $\mathcal{NP}$-hard, it is possible to approximate the optimal VC within a factor of 2 using an $O(m)$ algorithm based on determining a *maximal matching* of G [19].

B. Exploiting Knowledge of Router Mechanics and Traffic Flows: The Weak Vertex Cover Formulation

Using a VC of the network graph G to determine the set of nodes on which to run SNMP agents can obviously result in a

[1] Without loss of generality, we assume that all links of G are to be monitored. If only the edges in $E' \subset E$ are of interest, then G is understood to be the network subgraph spanned by E'.

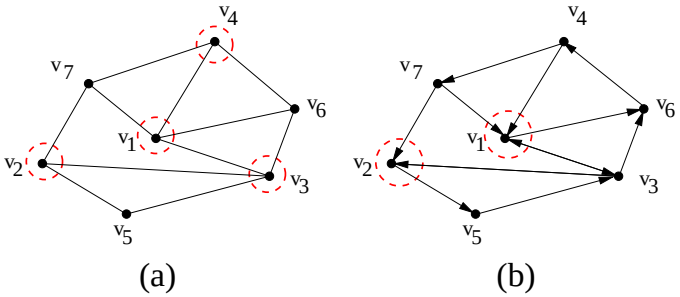

Fig. 1. (a) Network graph G and a minimum VC $S = \{v_1, v_2, v_3, v_4\}$. (b) A directed version of G and a minimum *Weak VC* $S = \{v_1, v_2\}$.

substantial reduction in the number of activated SNMP agents to monitor link-bandwidth usage in G. Nevertheless, it is possible to do even better by exploiting knowledge of the traffic flows in the network and the mechanics of packet forwarding. To simplify the exposition, we start by describing our novel problem formulation and algorithmic solutions assuming a *directed* network graph model G, with the direction of each link capturing the flow of data packets into or out of each router node. We then demonstrate how our results can be extended to the more realistic scenario of an undirected graph model.

B.1 The Weak Vertex Cover Problem for Directed Graphs

Consider a router v in the directed network graph G and let I_v (O_v) denote the set of incoming (resp., outgoing) edges incident to v in G. The key observation here is that, each such router v satisfies a *flow-conservation law* that, simply put, states that the sum of the traffic flowing into v is approximately the same as the sum of the traffic flowing out of v. More formally, the flow-conservation law for a *non-leaf* node [2] v can be stated as the following equation:

$$\sum_{e_i \in I_v} bw(e_i) - \sum_{e_j \in O_v} bw(e_j) \approx 0. \quad (1)$$

Note that, in practice, the above flow-conservation equation holds only approximately, since there can be (a) traffic directed to/from the router (e.g., OSPF protocol exchanges, management traffic, and ARP queries), (b) multicast traffic that is replicated along many output interfaces, and (c) delays and dropped packets in the router (under certain extreme congestion conditions). We believe, however, that these are infrequent conditions for routers in an ISP network that comprise only a very small proportion of the overall observed data traffic. Therefore, given a sufficiently large monitoring period, we expect the flow-conservation equation at each router to be approximately correct. Several measurements over backbone routers in Lucent's network have corroborated our expectations showing that flow conservation holds with a relative error that is consistently below 0.05% [18].

The importance of the flow-conservation law for network monitoring lies in the observation that we no longer need to ensure that all edges of a router are "covered" by an SNMP agent: *if a router has k links incident on it and the bandwidth utilization of $k-1$ of the links is known, then the bandwidth utilization of the remaining link can be derived from the flow-conservation equation for that router*. This observation leads to a novel vertex-covering formulation, termed *Weak Vertex Cover*.

Definition III.1: **[Weak Vertex Cover]** Given a directed network graph G, we define a set S of nodes to be a *Weak Vertex Cover of G* if after initially marking each node in S as covered, it is possible to mark every node in G by iteratively performing the following two steps: (1) Mark every edge in G that is incident on a covered node as covered; and, (2) For any non-leaf node v in G, if $deg(v) - 1$ of the edges incident to v are marked covered, then mark vertex v as covered. ∎

Based on the law of flow conservation, it is obvious that activating SNMP agents on a Weak VC for G is sufficient to derive the bandwidth usage on every link in the network graph. Thus, given flow conservation at each router of G, our efficient link-bandwidth monitoring problem becomes equivalent to determining a *minimum Weak VC* for G. For example, Figure 1(b) depicts a directed network G and the two nodes corresponding to a minimum Weak VC of G.

Note that every VC of a graph G is trivially also a Weak VC of G, but not necessarily a minimal one. (Figure 1 again provides a good example.) In fact, there are graphs for which the size of an optimal VC is arbitrarily larger than that of a Weak VC for the same graph (consider, for example, a long directed chain). Thus, exploiting the flow-conservation law can substantially improve the SNMP-monitoring overhead over a simple VC approach.

To the best of our knowledge, our Weak VC formulation represents a novel optimization problem that has not been studied in earlier research on combinatorial or graph algorithms. Unfortunately, as the following theorem shows, it is highly unlikely that we can find a minimum Weak VC in an efficient manner.

*Theorem III.1:*Given a directed graph G, discovering a minimum Weak VC is $\mathcal{NP}$-hard. ∎

A Near-Optimal Heuristic for Weak VC. An alternative way of viewing our Weak VC formulation is as follows. The law of flow conservation for every (non-leaf) router in G provides us with additional knowledge for the link-bandwidth unknowns ($bw(e_i)$) in the form of a *linear system of equations* that we can exploit to determine the values for all $bw(e_i)$'s. The problem then is to determine a minimum subset of nodes $S \subseteq V$ such that, when the $bw(e_i)$'s incident to all nodes in S are determined, the linear system of flow-conservation equations *can be solved* for all the remaining $bw(e_i)$'s. We now present a *provably near-optimal* greedy heuristic for Weak VC that is motivated from the above-stated formulation.

Let $\mathbf{A} \cdot \vec{bw} = \vec{b}$ denote the $n \times m$ linear system of flow-conservation equations corresponding to the (non-leaf) nodes in G (Equation (1)). (Without loss of generality, we assume that n is the number of *non-leaf* nodes in G.) Note that, initially, $\vec{b} = \vec{0}$ (i.e., the zero n-vector), but this is not necessarily the case in the later stages of our algorithm where some unknowns may have been specified by the routers selected in the cover. Also, let $rank(\mathbf{A})$ denote the rank of matrix $\mathbf{A}$, i.e., the number of *linearly independent* flow-conservation equations in the system.

Note that, if $rank(\mathbf{A}) \geq m$ then the linear system of flow-conservation equations can be directly solved to determine the values for all unknown link bandwidths $bw(e_i)$, which obviously means that no nodes need to be selected for monitoring. Otherwise, the *minimum required* number of link-bandwidth variables that need to be specified in order to make the flow-

[2] For simplicity, we assume that all links crossing the ISP network boundary are terminated by distinct leaf nodes in G.

conservation system solvable is exactly $m - rank(\mathbf{A})$. Selecting a node v to run an SNMP agent, means that all link-bandwidth variables attached to v become known and the flow-conservation equation for v becomes redundant. Thus, the original $n \times m$ system of flow-conservation equations is reduced to a $(n-1) \times (m - deg(v))$ system, where $deg(v)$ is the degree of node v in G.

Consider step k of the node-selection process (i.e., after enabling SNMP at $k-1$ selected nodes of G) and let G_k denote the network graph after the selected $k-1$ nodes (and incident edges) are removed from G. Also, let $\mathbf{A}_k \cdot \vec{bw}_k = \vec{b}_k$ denote the $n_k \times m_k$ system of flow-conservation equations for G_k. Finally, $l_k = m_k - rank(\mathbf{A}_k)$ denotes the *minimum number* of link variables that need to be (directly) specified so that the remainder network graph G_k is fully covered (i.e., the flow-conservation system for G_k becomes solvable). Our greedy algorithm for Weak VC, termed GREEDYRANK, selects at each step the node v that results in the *maximum possible reduction* in the minimum number of link variables required to make the flow-conservation equations solvable. That is, we select the node that maximizes the difference $l_k - l_{k+1}$. More formally, if we let $\mathbf{A}_k - \{v\}$ denote the $(n_k - 1) \times (m_k - deg(v))$ matrix resulting from the deletion of the flow-conservation equation for v and all its attached link variables from $\mathbf{A}_k$, then the GREEDYRANK strategy can be stated as depicted in Figure 2.

Algorithm GREEDYRANK(G)
Input: $G = (V, E)$ is the directed network graph comprising n nodes and m links.
Output: $S \subseteq V$ a Weak VC of G.

1) Let $\mathbf{A} \cdot \vec{bw} = \vec{b}$ denote the $n \times m$ linear system of flow-conservation equations for G;
2) Set $S = \{\}$, $k = 1$, $n_1 = n$, $m_1 = m$, $\mathbf{A}_1 = \mathbf{A}$, $G_1 = G$;
3) **while** $l_k = m_k - rank(\mathbf{A}_k) > 0$ **do**
4) Set v_{max} =*nil* and $maxReduction = 0$;
/* find node v that maximizes the reduction in the required number of variables */
5) **for each** node v in G_k **do**
6) $\delta_k(v) = l_k - (m_k - deg(v) - rank(\mathbf{A}_k - \{v\}))$
7) **if** $\delta_k(v) > maxReduction$ **then**
8) Set $v_{max} = v$ and $maxReduction = \delta_k(v)$;
9) Set $S = S \cup \{v_{max}\}$, $\mathbf{A}_{k+1} = \mathbf{A}_k - \{v_{max}\}$, $G_{k+1} = G_k - \{v_{max}\}$, $n_{k+1} = n_k - 1$, $m_{k+1} = m_k - deg(v_{max})$, and $k = k + 1$;

Fig. 2. Finding a Near-Optimal Weak Vertex Cover.

The following theorem bounds the worst-case behavior of our GREEDYRANK algorithm.

*Theorem III.2:*Algorithm GREEDYRANK returns a solution to the Weak VC problem that is guaranteed to be within a factor of $1 + \ln(m - rank(\mathbf{A}))$ of the optimal solution, where $\mathbf{A}$ is the coefficient matrix of the n flow-conservation equations in G. ∎

Time Complexity. GREEDYRANK requires repeated matrix-rank computations in order to determine the "locally-optimal" node to place in the cover at each step. However, as we show in the full version of this paper [18], the specific form of the coefficient matrix $\mathbf{A}$ allows us to reduce matrix-rank computation to a simple search for (undirected) connected components in G, which can be performed in $O(\max\{n, m\})$ time. Consequently, the worst-case running time of our GREEDYRANK algorithm can be shown to be only $O(n^2 \cdot \max\{n, m\})$.

B.2 Exploiting Knowledge of Network Flows

So far, our Weak VC formulation makes use of the law of flow conservation on each router but it does not exploit knowledge of traffic flows in the network when trying to estimate link-bandwidth usage. This flow information essentially consists of the paths along which packets get routed in the network and can be computed using routing protocol control information (e.g., the link-state database in OSPF or label switched paths in MPLS). In this section, we demonstrate how knowledge of the traffic flows in our directed network graph G can be exploited in conjunction with flow conservation to further reduce the required SNMP overhead for monitoring link bandwidth.

Consider a router v in G and let E_v, $\mathcal{F}_v$ denote the (sub)sets of links incident on v and packet flows routed through v, respectively. We can always uniquely *partition* E_v into a maximal collection of $k(v) \geq 1$ subsets $E_v^1, \ldots, E_v^{k(v)}$ such that each flow $F \in \mathcal{F}_v$ only involves (a pair of) links in a single subset E_v^i, for some i. We say that such a partitioning of the links in E satisfies the *non-overlapping flow property*. An example of this flow-based link partitioning (with $k(v) = 2$) for a node v is depicted in Figure 3. (In the worst case $k(v) = 1$, i.e., a node's links cannot be partitioned into non-overlapping flows and, thus, they all belong to the same partition.) The key observation here is that the law of flow conservation in fact holds *for each individual link partition* E_v^i, $i = 1, \ldots, k(v)$; thus, node v can be marked as covered as long as $|E_v^i| - 1$ links in E_v^i are covered for each $i = 1, \ldots, k(v)$. This essentially means that we can infer the link-bandwidth utilization on each link incident to v based on knowing the bandwidth usage on only $|E_v| - k(v)$ links of v ($k(v) \geq 1$). (Note, of course, that these $|E_v| - k(v)$ links have to satisfy the condition outlined above; that is, only one link may be left unspecified in each partition E_v^i, $i = 1, \ldots, k(v)$.) As an example, knowing the bandwidth usage on the three outgoing links in Figure 3 is sufficient to infer the bandwidth load on the two incoming links. This leads us to a generalized formulation of our Weak VC problem.

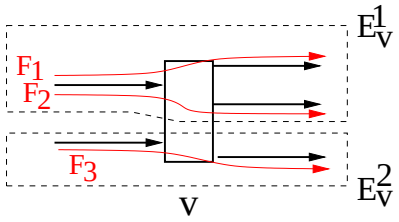

Fig. 3. Partitioning v's edges into $k(v) = 2$ partitions (E_v^1 and E_v^2) satisfying the non-overlapping flow property.

Definition III.2: **[Partitioned Weak Vertex Cover]** Given a directed network graph G and a partitioning of the links in G that satisfies the non-overlapping flow property, we define a set S of nodes to be a *Partitioned Weak VC of G* if after initially marking each node in S as covered, it is possible to mark every node in G by iteratively performing the following three steps: (1) Mark every link in G that is incident on a covered node as covered; (2) For any node v in G, if in *any* link partition E_v^i ($i = 1, \ldots, k(v)$) there are $|E_v^i| - 1$ links marked covered then

mark the remaining link in E_v^i as covered; and, (3) For any node v in G, if in *every* link partition E_v^i ($i = 1, \ldots, k(v)$) there are at least $|E_v^i| - 1$ links marked covered then mark vertex v as covered. ∎

Generating the maximal link partitioning of E_v (for each v in G) that satisfies the non-overlapping flow property is fairly straightforward. The idea is to start by placing each link in its own partition and iteratively "merge" partitions that share flows [18]. Based on the law of flow conservation for each link partition, it is easy to see that, activating SNMP agents on a Partitioned Weak VC for G is sufficient to derive the bandwidth usage on every link in the network graph. Thus, given traffic-flow information in the network, our low-overhead link-bandwidth monitoring problem becomes equivalent to determining a *minimum Partitioned Weak VC* for G. This problem is clearly $\mathcal{NP}$-hard (a generalization of Weak VC); however, a version of our GREEDYRANK algorithm can be used to give a fast approximate solution with a guaranteed logarithmic worst-case performance bound. (Note that, for Partitioned Weak VC, removing a node v from the linear system of flow-conservation equations means that all $k(v)$ equations corresponding to v are removed.) Due to lack of space, the full details can be found in [18].

*Theorem III.3:*For the Partitioned Weak VC problem, algorithm GREEDYRANK runs in time $O(n^2 \cdot \max\{n, m\})$ and returns a solution that is guaranteed to be within a factor of $1 + \ln(m - rank(\mathbf{A}))$ of the optimal solution, where $\mathbf{A}$ is the coefficient matrix of the $\sum_v k(v)$ flow-conservation equations in G. ∎

B.3 Extension to Undirected Network Graphs

Our discussion of link-bandwidth monitoring has so far focused on the case of a directed network graph model G, where packet traffic on each physical link is uni-directional and known beforehand. In general, physical network links are bi-directional with data packets flowing both to and from routers on the same link. We now briefly describe how our results and algorithmic techniques extend to this more general and realistic scenario of an undirected network graph model.

The basic idea is to "expand" the network graph G into a directed graph G_d by modeling each bi-directional physical link in G as *two* directed edges (in opposing directions) in G_d, thus capturing both directions of packet flow for each link. Of course, one (or both) of the directed edges for a link can be left out of the model if we know that it is not being used for actual traffic in the network, e.g., based on the knowledge of traffic flows. The flow-conservation law (Equation (1)) then holds for each router in G_d and its directed in- and out-links created in this manner. Thus, our solutions for Weak VC and Partitioned Weak VC can be directly applied on this "expanded" directed network graph G_d. More details can be found in [18].

C. Monitoring Flow-Bandwidth Utilization

Consider the (undirected) network graph G with a set of packet flows $\mathcal{F} = \{F_1, F_2, \ldots\}$ routed through its nodes. Enabling RMON or NetFlow on a router v of G enables the bandwidth utilization for the set $\mathcal{F}_v$ of all flows routed through v to be measured, i.e., v *directly covers* all the flows in $\mathcal{F}_v$. It is straightforward to see that the problem of determining a minimum subset of RMON/NetFlow-enabled routers such that every flow in $\mathcal{F}$ is directly covered is essentially an instance of the $\mathcal{NP}$-hard *Set Cover* problem [15]. Thus, a greedy Set-Cover heuristic can be used to return an approximate solution that is guaranteed to be within $1 + \ln|\mathcal{F}|$ of the optimal in $O(n \cdot |\mathcal{F}|)$ time [18].

Exploiting Link Bandwidth Information for Covering Flows. The performance overhead imposed by tools like RMON or NetFlow on routers and network traffic is typically significantly higher than that of SNMP, mainly due to their much finer granularity of data collection. The problem, of course, is that SNMP agents cannot collect and provide traffic data at the required granularity of packet flows – only aggregate information on link-bandwidth utilization can be obtained through SNMP.

The crucial observation here is that knowledge of aggregate link bandwidths (obtained via SNMP) provides us with a system of *per-link* linear equations on the unknown flow-bandwidth utilizations that can be exploited to significantly reduce the number of RMON/NetFlow probes required for monitoring flow-bandwidth usage. (Each such equation basically states that the aggregate link bandwidth is equal to the sum of the bandwidth utilizations of the flows traversing that link [18].) The resulting problem is similar to Weak VC, except that we are interested in a minimum subset S of routers such that determining the bandwidth utilization of flows passing through routers in S renders the system of per-link equations solvable. The following theorem establishes the intractability of this optimization problem and the near-optimality of our general GREEDYRANK strategy. Due to lack of space, the full details can be found in [18].

Theorem III.1: Given knowledge of the aggregate link-bandwidth utilizations in G, the problem of determining a minimum subset S of routers such that enabling RMON/NetFlow on every $v \in S$ allows the determination of the flow-bandwidth utilizations for every flow in $\mathcal{F}$ is $\mathcal{NP}$-hard. Further, an appropriately modified version of our GREEDYRANK strategy returns a solution to this problem that is guaranteed to be within a factor of $1 + \ln(|\mathcal{F}| - rank(\mathbf{A}))$ of the optimal, where $\mathbf{A}$ is the $m \times |\mathcal{F}|$ coefficient matrix of the per-link flow-bandwidth equations in G. This approximation factor is the best possible (assuming $\mathcal{P} \neq \mathcal{NP}$). ∎

IV. MONITORING NETWORK LATENCY

We next turn our attention to the problem of measuring round-trip latencies for a set of network paths P in the (undirected) network graph G, where each *path* is a sequence of adjacent links in G. Such network latency measurements are crucial for providing QoS guarantees to end applications (e.g., voice over IP), traffic engineering, ensuring SLA compliance, fault and congestion detection, performance debugging, network operations, and dynamic replica selection on the Web.

Most previous proposals for measuring round-trip times of network paths rely on *probes*, which are simply ICMP echo request packets. Existing systems typically belong to one of two categories. The first category includes systems like WIPM [20], AMP [21] and IDMaps [3] that deploy special *measurement servers* at strategic locations in the network. Round-trip times between each pair of servers are measured using probes, and these times are subsequently used to approximate the latency of arbitrary network paths. The measurement server approach, while popular, suffers from the following two drawbacks. First, the cost of deploying and managing hundreds of geographically distributed servers can be significant due to the required hard-

ware and software infrastructure as well as the human resources. Second, the accuracy of the latency measurements is highly dependent on the number of measurement servers. With few servers, it is possible for significant errors to be introduced when round-trip times between the servers are used to approximate arbitrary path latencies. The second category of tools for measuring path latencies include `pathchar` [6] and *skitter* [22]. Both tools measure the round-trip times for paths originating at a small set of sources (between one and ten) by sending probes with increasing TTL values from each source to a large set of destinations. A shortcoming of these tools is that they can only measure latencies of a limited set of paths that begin at one of the sources from which ICMP probes are sent.

In this section, we present our *probing-based* technique that alleviates the drawbacks of previous methods. In our approach, path latencies are measured by transmitting probes from a single point-of-control (i.e., the NOC). Consequently, since our technique does not require special instrumentation to be installed in the network, it is cost-effective and easy to deploy. Further, unlike existing approaches, our method allows for latencies of an arbitrary set of network paths P to be measured exactly, and is thus both accurate and flexible. Our schemes achieve this by exploiting the ability within IP to *explicitly* route packets using either source routing or encapsulation of "IP in IP". We demonstrate that, for measuring the latency of a given set of paths P, there exist a wide range of probing strategies that impose varying amounts of load on the network infrastructure. While the problem of selecting the optimal set of probes that minimizes the network bandwidth consumed is $\mathcal{NP}$-hard, we show that this problem can be mapped to the well-known *Facility Location Problem* (FLP) for which efficient approximation algorithms with guaranteed performance ratios have been proposed in the literature [16], [23].

A. Overview and Problem Formulation

In our approach for measuring latency, explicitly routed probes are transmitted along paths originating at the NOC (i.e., node v_0). (We discuss source routing and IP encapsulation, the two mechanisms within IP for controlling the path traversed by a packet in more detail in Section IV-D.) The round-trip latency of a single path p_i is measured by sending the following two probe packets:

1. The first probe packet is sent from v_0 to one of the end nodes, say v_i, along the shortest path β_i between them in G. The probe then returns to v_0 along the reverse of β_i.
2. The second probe packet is sent from v_0 to the other end node v_j of p_i (via v_i) along the path $\beta_i \cdot p_i$. The probe then returns to v_0 along the reverse of $\beta_i \cdot p_i$.

The round-trip latency of path p_i is computed as the difference of the round-trip times of the two probes traversing the paths β_i and $\beta_i \cdot p_i$.

In the remainder of this paper, we will represent each probe by the forward path traversed by it from v_0 since the return path is symmetric to the forward path (in the reverse direction). Note that the first node of each probe is always v_0. Also, in this paper, we will not consider complex probing techniques for measuring latency in which probes follow arbitrary paths which cannot be decomposed into symmetric forward and reverse path segments[3].

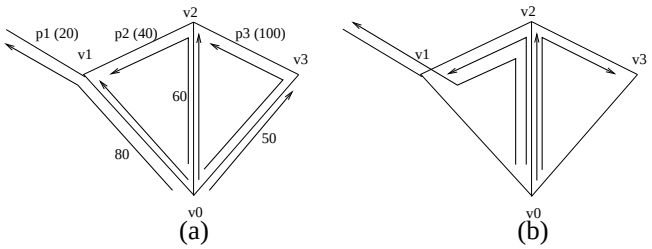

Fig. 4. Sets of Probes for Measuring Latency of Paths

In order to measure the latency of a set of paths P, we need to employ a set of probes S such that for each path $p_i \in P$, S contains a pair of probes α_i and $\alpha_i \cdot p_i$. We refer to α_i as the *base* probe for p_i and to $\alpha_i \cdot p_i$ as the *measurement* probe for p_i. Further, we refer to probes in S that are not measurement probes for any path in P as *anchor* probes and to the last node visited by an anchor probe as its *anchor* node.

There are a number of different sets of probes that can be used to measure the latency of P. One obvious choice for S is the set that contains, for each path $p_i \in P$, the following two probes: β_i and $\beta_i \cdot p_i$ (assuming that v_i is the end node of p_i that is closest to v_0). However, as the following example illustrates, there are a number of other possibilities for set S, several of which contain a smaller number of probes and traverse fewer links.

Example IV.1: Consider the network graph shown in Figure 4(a) containing the set of paths $P = \{p_1, p_2, p_3\}$, where $|p_1| = 20$, $|p_2| = 40$, $|p_3| = 100$, $|\beta_1| = 80$, $|\beta_2| = 60$, and $|\beta_3| = 50$. The obvious set of probes S_1 for measuring the latency of P is illustrated in Figure 4(a), where the measurement of each path is optimized independently by sending the base probe to the end node closest to v_0. This results in a distinct pair of probes, β_i and $\beta_i \cdot p_i$, for each path p_i, which requires a total of 6 probes and 540 traversed links. Figure 4(b) illustrates a different set of probes S_2 for measuring P that is optimal with respect to both the number of probes as well as the number of traversed links. In S_2, paths p_2 and p_3 share the same base probe β_2, and the measurement probe for p_2 (i.e., $\beta_2 \cdot p_2$) also serves as the base probe for p_1. (Note that β_2 is the *only anchor probe* in S_2, whereas S_1 contains three distinct anchor probes ($\beta_1, \beta_2, \beta_3$).) This sharing of probes among paths reduces the number of probes from 6 to 4. Although both paths p_1 and p_3 are measured with longer measurement probes in S_2 ($\beta_2 \cdot p_2 \cdot p_1$ and $\beta_2 \cdot p_3$) than in S_1, this overhead is offset by the savings due to the sharing of probes in S_2, thereby resulting in an overall reduction from 540 to 440 traversed links. ∎

Ideally, we would prefer a set S of probes that traverses as few links as possible to measure P. This is because the total number of links traversed by the probes in S is a good measure of the additional load that the probes impose on network links. Minimizing this additional network traffic due to probes is extremely important, since we need to monitor path latencies continuously, causing probes to be transmitted frequently (e.g., every fifteen minutes). Thus, our efficient latency-monitoring

[3]One such complex probing technique for measuring the latency of a path p_i would be to send a pair of probes – the first probe makes a round-trip from v_0 to an internal node, say v_k, in p_i; the second probe travels from v_0 to one of the end nodes of p_i via v_k, and then to the other end node of p_i, and finally back to v_0 via v_k.

problem can be formally stated as follows.

Problem Statement [Low-Overhead Path-Latency Monitoring]: Given a set of paths P, compute a set of probes S such that (1) S measures the latency of P; that is, for every path $p_i \in P$, S contains a pair of probes α_i and $\alpha_i \cdot p_i$, and (2) S is *optimal*; that is, the total number of links traversed by probes in S is minimum. ∎

In the following subsection, we address the above problem of computing the optimal set of probes for measuring the latency of paths in P. We assume that for any pair of paths p_i and p_j in P, p_i is not a prefix (or suffix) of p_j. The reason for this assumption will become clear in the next subsection. Note, however, that this assumption is not restrictive since, if p_i is a prefix of $p_j = p_i \cdot p_l$, then p_j can be split into two non-overlapping path segments p_i and p_l, and its latency can be computed as the sum of the latencies of p_i and p_l.

B. Computing an Optimal Set of Probes

As illustrated in Example IV.1, a naive approach that adds to S the optimal pair of base and measurement probes for each path in P considered independently may not result in the optimal set of probes. This is because (1) measurement probes for multiple paths can share a common base probe, and (2) the measurement probe for one path can serve as a base probe for an adjoining path. Thus, more sophisticated algorithms are needed for computing an optimal solution. Unfortunately, as the following theorem states, the problem of computing the optimal set of probes is $\mathcal{NP}$-hard even if every path in P is restricted to be a single link.

*Theorem IV.1:*Given a graph G and a set of paths $P \subset E$, the problem of computing the optimal set of probes to measure the latency of P is $\mathcal{NP}$-hard. ∎

In the following, we map the problem of computing the optimal set of probes to the *Facility Location Problem* (FLP). Since efficient polynomial-time algorithms for approximating the FLP exist in the literature, these can then be utilized to compute a near-optimal set of probes.

Before we present our FLP reduction, we develop some additional notation. For a path α_i, we denote by $f(\alpha_i)$ and $l(\alpha_i)$ the first and last node of α_i, respectively. Further, $G_p = (V_p, E_p)$ denotes the undirected, distance-weighted graph induced by P; thus, $V_p = \{f(p_i), l(p_i) \mid p_i \in P\}$ is the set of all the end nodes of the paths in P, and $E_p = \{(f(p_i), l(p_i)) \mid p_i \in P\}$ is the set of edges between the end nodes such that corresponding to every path in P, there is an edge in G_p connecting its two end nodes. (Note that G_p is actually a *multigraph*, since there may be multiple edges between a pair of nodes in G_p.) Each edge in G_p is labeled with its corresponding path, say p_i, in G, and has an associated *weight* equal to $|p_i|$. For a pair of nodes v_i and v_j in G_p, we denote by $\delta_{i,j}$ the shortest path (with respect to the sum of edge weights) from v_i to v_j in G_p. Essentially, $\delta_{i,j}$ is the path from v_i to v_j in the shortest path tree rooted at v_i in G_p. Note that, since every edge in G_p corresponds to some path in P, $\delta_{i,j}$ can be viewed as a concatenation of paths in P . Finally, we use $|\delta_{i,j}|$ to denote the sum of edge weights for $\delta_{i,j}$ in G_p.

We are now in a position to characterize the composition of sets of probes. A set S of probes for measuring the latency of paths in P consists of the following two disjoint subsets:
1. A set of anchor probes S_a (corresponding to anchor nodes $V_a \subseteq V$), and

---

Algorithm OPTIMALPROBES(G, P, V_a)
Input: $G = (V, E)$ is a network graph.
P = A set of paths whose latency is to be measured.
V_a = A set of anchor nodes.
Output: S_m = A set of measurement probes that is optimal with respect to V_a and P.
1) $S_m = \{\}$;
2) **for each** $p_i \in P$ **do**
3) Let $v_j \in V_a$ and $v_k \in \{f(p_i), l(p_i)\}$ such that $|\beta_j \cdot \delta_{j,k}|$ is minimum (in case of a tie between nodes v_j and v_l in V_a, the node with the smaller index is chosen);
4) $S_m = S_m \cup \{\beta_j \cdot \delta_{j,k} \cdot p_i\}$;

Fig. 5. Finding an Optimal Set of Probes for Anchor Nodes V_a.

---

2. A set of measurement probes S_m for measuring the latency of paths in P.
Since the shortest possible anchor probe in S_a corresponding to each anchor node $v_i \in V_a$ is β_i, $S_a = \{\beta_i \mid v_i \in V_a\}$. In S_m, there is a separate measurement probe for each path in P (since for any pair of paths $p_i, p_j \in P$, p_i cannot be a prefix of p_j). Further, every measurement probe in S_m is a concatenation of a single anchor probe and one or more paths in P. The shortest possible measurement probe in S_m to measure the latency of path p_i has length equal to $\min_{v_j \in V_a, v_k \in \{f(p_i), l(p_i)\}} |\beta_j \cdot \delta_{j,k} \cdot p_i|$. Here, the minimization over $|\beta_j \cdot \delta_{j,k}|$ essentially captures the shortest possible path from v_0 to one of the end nodes v_k of p_i that begins with an anchor probe β_j followed by paths in P. Thus, for a given set of anchor nodes, it is possible to compute the optimal set of measurement probes S_m for measuring P. Algorithm OPTIMALPROBES in Figure 5 computes this optimal set S_m for a given V_a and P. The computed set S_m is optimal since, for each path $p_i \in P$, OPTIMALPROBES adds to S_m the measurement probe containing the smallest number of links. In addition, we can prove that the set $S_a \cup S_m$ measures the latency of all paths in P. For this, we need to show that, for every path p_i, the base probe $\beta_j \cdot \delta_{j,k}$ is either in S_a or is added to S_m. If $\delta_{j,k} = \epsilon$, then the base probe β_j is in S_a. Otherwise, if p_l is the final path in $\delta_{j,k}$, then $\beta_j \cdot \delta_{j,k}$ is the measurement probe for p_l with the smallest number of links and is thus added to S_m. Thus, the set of probes $S_a \cup S_m$ measures the latency of P.

*Theorem IV.2:*Given a set of anchor nodes V_a and paths P, Algorithm OPTIMALPROBES computes an optimal set of measurement probes S_m such that $S_a \cup S_m$ measures the latency of P. ∎

From Theorem IV.2, it follows that for a given set of anchor nodes V_a, the set $S = S_a \cup S_m$ is the optimal set of probes for measuring P among sets for which S_a is the set of anchor probes. Further, the number of links traversed by S is:

$$\sum_{v_j \in V_a} |\beta_j| + \sum_{p_i \in P} \min_{v_j \in V_a, v_k \in \{f(p_i), l(p_i)\}} |\beta_j \cdot \delta_{j,k} \cdot p_i|. \quad (2)$$

Thus, if V_a is a set of anchor nodes for an *optimal* set of probes, then $S_a \cup S_m$ (where S_m is computed by OPTIMALPROBES) is an optimal set of probes for measuring P; that is, $S_a \cup S_m$ minimizes the value of Equation (2). As a result, we have transformed the problem of computing the optimal set of probes to that of computing a set of anchor nodes V_a that minimizes Equa-

tion (2). Once V_a is known, algorithm OPTIMALPROBES can be used to compute the measurement probes S_m such that $S_a \cup S_m$ is the optimal set of probes.

The above minimization problem maps naturally to the *Facility Location Problem* (FLP) [16], [23]. The FLP is formulated as follows: Let C be a set of clients and J be a set of facilities such that each facility "serves" every client. There is a cost $c(j)$ of "choosing" a facility $j \in J$ and a cost $d(j,i)$ of serving client $i \in C$ by facility $j \in J$. The problem definition asks to choose a subset of facilities $F \subseteq J$ such that the sum of costs of the chosen facilities plus the sum of costs of serving every client by its closest chosen facility is minimized; that is, $\min_{F \subseteq J}\{\sum_{j \in F} c(j) + \sum_{i \in C} \min_{j \in F} d(j,i)\}$.

The problem of computing the set of anchor nodes V_a that minimizes Equation (2) can be mapped to FLP as follows: Let C be the set of paths P and J be the set of candidate anchor nodes V_p. The cost of choosing a facility j, $c(j)$, is $|\beta_j|$, the length of the shortest path from v_0 to v_j. The cost of serving client p_i from facility v_j, $d(j,i)$, is $\min_{v_k \in \{f(p_i), l(p_i)\}}\{\ |\beta_j| + |\delta_{j,k}|\ \}$ which is the sum of the lengths of β_j and the shortest path from v_j to one of the end nodes of p_i in G_p. Thus, the set F computed for the FLP corresponds to our desired optimal set V_a of anchor nodes.

The FLP is $\mathcal{NP}$-hard; however, it can be reduced to an instance of the *Set Cover* problem and then approximated within a factor of $O(\log |C|)$ with a running-time complexity of $O(|C|^2 \cdot |J|)$ [16]. Thus, we can compute a provably near-optimal set S of probes for measuring paths in P by first using Hochbaum's FLP heuristic [16] to find a near-optimal set of anchor nodes V_a (in $O(|P|^2 \cdot |V_p|)$ time), and then running OPTIMALPROBES to find the optimal set of measurement probes S_m for V_a. The set of probes $S = S_a \cup S_m$ is then guaranteed to be within $O(\log |P|)$ of the optimal solution for measuring P [18].

C. Minimizing the Number of Probes

Suppose that instead of minimizing the number of traversed links, we are interested in computing the set S with the minimum number of probes. In this case, it is possible to compute the optimal set of probes by invoking algorithm OPTIMALPROBES with the set of paths P whose latency is to be measured and the set V_a that contains one (arbitrary) node from each connected component in G_p. The final set $S = S_a \cup S_m$ contains one anchor probe per connected component in G_p and one measurement probe per path, which is optimal with respect to the number of probes.

D. Implementation Issues

Our approach for measuring latency is highly dependent on being able to explicitly route probe packets along specific paths. Loose source routing and encapsulation of IP in IP are two mechanisms for controlling routes followed by packets. We prefer encapsulation over loose source routing due to the following reasons [24]. First, Internet routers exhibit performance problems when forwarding packets that contain IP options, including the IP source routing option. Second, the source routing option is frequently disabled on Internet routers due to security problems associated with its use. Finally, IP allows for at most 40 bytes of options, which restricts the number of IP addresses through which a packet can be routed using source routing to be no more than 10.

While encapsulation addresses some of the problems with source routing, unwrapping the header in encapsulated packets still incurs overhead at routers and encapsulated packets are typically larger than source routed packets. Both processing overhead and packet sizes can be reduced significantly by using as few headers as possible in each probe packet. We can achieve this by splitting the path for a probe packet into maximal disjoint path segments, such that each path segment is consistent with the route computed by the underlying routing protocol (e.g., OSPF). Then the probe packet can be routed along the path by using one header per path segment that contains the IP address of the endpoint of the segment that is not shared with the previous segment. Note that the final measured round-trip times must be adjusted to account for the overhead of processing the encapsulated packets at intermediate routers.

V. SIMULATION RESULTS

In this section, we present simulation results comparing the performance of the various algorithms that we have developed for both the link-bandwidth and path-latency measurement problems. The main objective of the simulation results is to demonstrate that our proposed algorithmic solutions are not only theoretically sound with good guaranteed worst-case bounds but they also give significant benefits over naive solutions in practice (i.e., on the average) for a wide variety of realistic network topologies. The simulations are based on network topologies generated using the Waxman Model [17], which is a popular topology model for networking research (e.g., [4]). Different network topologies are generated by varying three parameters: (1) n, the number of nodes in the network graph; (2) α, a parameter that controls the density of short edges in the network; and (3) β, a parameter that controls the average node degree.

A. Bandwidth Measurement

For the link-bandwidth measurement problem, we compare the performance of three algorithms: the maximal matching heuristic for simple VC (Sec. III-A), and two algorithms based on our Weak VC formulation – a variant of the maximal matching heuristic and our GREEDYRANK algorithm (Sec. III-B.1). Our maximal matching variant for Weak VC basically ensures that all transitively-specified edges (based on flow conservation) are eliminated from G whenever a new edge enters the matching. The comparison is in terms of the number of nodes that need to run SNMP in order to measure the bandwidth of each link in the generated network graphs[4]. We denote the number of SNMP activations for these algorithms by N^{vc}_{match}, N^{wvc}_{match}, and N^{wvc}_{rank}, respectively.

Table II presents one set of simulation results; we have obtained similar results for other parameter settings. The first column in the table represents the average degree of the nodes in the generated network graph (which increases with larger values of β). Our results indicate that GREEDYRANK is the clear winner, reducing the number of SNMP activations by as much as 67% over the naive,"brute-force" approach, and as much as 35% over its closest matching-based competitor.

[4]For the bandwidth measurement simulations, each undirected graph generated by the Waxman model is converted into a directed graph by randomly fi xing the direction of each of its edges.

Avg. Degree	N^{vc}_{match}	N^{wvc}_{match}	N^{wvc}_{rank}	$\frac{N^{wvc}_{rank}}{n}$
4.4	387	255	165	0.33
8.6	441	372	254	0.51
12.6	453	408	307	0.61
16.9	466	431	334	0.67

TABLE II

COMPARISON OF LINK-BANDWIDTH MEASUREMENT ALGORITHMS, $n = 500$, $\alpha = 0.4$, $\beta \in \{0.02, \ldots, 0.08\}$

B. Latency Measurement

For the latency measurement simulations, we compare the performance of two algorithms: the *naive* approach, where the optimal probes are computed independently for each path (Sec. IV-A), and our FLP-based approach (Sec. IV-B). We compare the performance of these algorithms in terms of both the total number of links traversed by the probe packets, denoted by L_x, as well as the number of probe packets transmitted, denoted by P_x, where $x \in \{naive, flp\}$.

For each network graph generated using the Waxman model, a random set of 20 paths (each with between 2 and 10 links) are considered. We vary the "topology" of the set of generated paths using a parameter n', which represents the number of end nodes that serve as starting points for the 20 generated paths. Thus, a smaller value of n' means that more paths are terminated by the same end node. The node representing the NOC is a randomly selected node that is not incident on any of the paths.

Table II presents one set of simulation results; we have observed similar trends for other parameter settings. The results indicate that our FLP-based heuristic is more effective than the naive approach in terms of both the total number of links traversed as well as the total number of probe packets transmitted.

n'	L_{naive}	L_{flp}	$\frac{L_{flp}}{L_{naive}}$	P_{naive}	P_{flp}	$\frac{P_{flp}}{P_{naive}}$
2	684	542	0.79	37	22	0.59
4	672	560	0.83	37	24	0.65
8	678	594	0.88	38	28	0.74
16	680	628	0.92	39	32	0.82

TABLE III

COMPARISON OF LATENCY MEASUREMENT ALGORITHMS, $n = 1000$, $\alpha = 0.2$, $\beta = 0.02$.

VI. CONCLUSIONS

In this paper, we have addressed the problem of efficiently monitoring bandwidth utilization and path latencies in IP networks. Unlike earlier approaches, our measurement architecture assumes a single point-of-control in the network (corresponding to the NOC) that is responsible for gathering bandwidth and latency information using widely-deployed management tools, like SNMP, RMON/NetFlow, and explicitly-routed IP probes. We have demonstrated that our measurement model gives rise to new optimization problems, most of which prove to be $\mathcal{NP}$-hard. We have also developed novel approximation algorithms for these optimization problems and proved guaranteed upper bounds on their worst-case performance. Finally, we have verified the effectiveness of our monitoring algorithms through a preliminary simulation evaluation.

Although this paper has focused on a single point-of-control measurement architecture, our approach is also readily applicable to a distributed-monitoring setting, where a number of NOCs/"monitoring boxes" have been distributed over a large network area with each NOC responsible for monitoring a smaller region of the network. Our algorithms can then be used to minimize the monitoring overhead within each individual region. The problem of optimal distribution and placement of NOCs across a large network can be formulated as a variant of the well-known *"k-center problem"* (with an appropriately-defined distance function) [4].

Acknowledgement: We would like to thank Amit Kumar for suggesting the $O(\max\{m, n\})$ rank-computation algorithm for GREEDYRANK.

REFERENCES

[1] W. Stallings, *"SNMP, SNMPv2, SNMPv3, and RMON 1 and 2"*, Addison-Wesley Longman, Inc., 1999, (Third Edition).
[2] "NetFlow Services and Applications," Cisco Systems White Paper, 1999.
[3] P. Francis, S. Jamin, V. Paxson, L. Zhang, D. F. Gryniewicz, and Y. Jin, "An Architecture for a Global Internet Host Distance Estimation Service," in *Proc. of IEEE INFOCOM'99*, March 1999.
[4] S. Jamin, C. Jin, Y. Jin, Y. Raz, Y. Shavitt, and L. Zhang, "On the Placement of Internet Instrumentation," in *Proc. of IEEE INFOCOM'2000*, March 2000.
[5] W. Theilmann and K. Rothermel, "Dynamic Distance Maps of the Internet," in *Proc. of IEEE INFOCOM'2000*, March 2000.
[6] V. Jacobsen, "pathchar – A Tool to Infer Characteristics of Internet Paths," April 1997, `ftp://ftp.ee.lbl.gov/pathchar`.
[7] A.B. Downey, "Using pathchar to Estimate Internet Link Characteristics," in *Proc. of ACM SIGCOMM'99*, August 1999.
[8] J.-C. Bolot, "End-to-End Packet Delay and Loss Behavior in the Internet," in *Proc. of ACM SIGCOMM'93*, September 1993.
[9] K. Lai and M. Baker, "Measuring Bandwidth," in *Proc. of IEEE INFOCOM'99*, March 1999.
[10] M. Cheikhrouhou, J. Labetoulle, "An Effi cient Polling Layer for SNMP," *Proc. 2000 IEEE/IFIP Network Operations & Management Symposium*, April 2000.
[11] Y. Yemini, G. Goldszmidt, S. Yemini, "Network Management by Delegation," *Proc. Intl Symposium on Integrated Network Management*, April 1991.
[12] D. Breitgand, D. Raz, Y. Shavitt, "SNMP GetPrev: An Effi cient Way to Access Data in Large MIB Tables," Bell Labs Tech. Memorandum, August 2000.
[13] V. Paxson, "Towards a Framework for Defi ning Internet Performance Metrics," in *Proceedings of INET'96*, 1996.
[14] S. Keshav, *"An Engineering Approach to Computer Networking"*, Addison-Wesley Professional Computing Series, 1997.
[15] M.R. Garey and D.S. Johnson, *"Computers and Intractability: A Guide to the Theory of NP-Completeness"*, W.H. Freeman, 1979.
[16] D.S. Hochbaum, "Heuristics for the Fixed Cost Median Problem," *Mathematical Programming*, vol. 22, pp. 148–162, 1982.
[17] B.M. Waxman, "Routing of Multipoint Connections," *IEEE Jrnl. on Selected Areas in Communications*, vol. 6, no. 9, pp. 1617–1622, December 1988.
[18] Y. Breitbart, C.-Y. Chan, M. Garofalakis, R. Rastogi, and A. Silberschatz, "Effi ciently Monitoring Bandwidth and Latency in IP Networks," Bell Labs Tech. Memorandum, July 2000.
[19] V. V. Vazirani, *"Approximation Algorithms"*, Springer-Verlag, 2000, (To appear).
[20] R. Caceres, N.G. Duffi eld, A. Feldmann, J. Friedmann, A. Greenberg, R. Greer, T. Johnson, C. Kalmanek, B. Krishnamurthy, D. Lavelle, P.P. Mishra, K.K. Ramakrishnan, J. Rexford, F. True, and J.E. van der Merwe, "Measurement and Analysis of IP Network Usage and Behaviour," *IEEE Communications Magazine*, pp. 144–151, May 2000.
[21] T. McGregor, H.-W. Braun, and J. Brown, "The NLANR Network Analysis Infrastructure," *IEEE Communications Magazine*, pp. 122–128, May 2000.
[22] Cooperative Association for Internet Data Analysis (CAIDA), `http://www.caida.org/`.
[23] M. Charikar and S. Guha, "Improved Combinatorial Algorithms for the Facility Location and k-Median Problems," in *Proc. of IEEE FOCS'99*, October 1999.
[24] C. Perkins, "IP encapsulation within IP," Internet RFC-2003 (available from `http://www.ietf.org/rfc/`), May 1990.