An Efficient Overlay Link Performance Monitoring Technique

Zhi Li[1], Lihua Yuan[2], and Prasant Mohapatra[2]

[1] Network Systems Engineering, AT&T
lizhi@cs.ucdavis.edu
[2] University of California, Davis
{lyuan@ece, prasant@cs}.ucdavis.edu

Abstract. Link performance monitoring is a common task required by different overlay networks. Current overlays typically let each node monitor links by itself, which is not scalable for large networks. Earlier improvement proposals either use a centralized approach or sacrifice measurement accuracy. This paper proposes *MONET*, a distributed overlay monitoring technique. Based on the proposed *X-Set* concept, MONET enables peer cooperation so that each node performs a minimum amount of measurement but can deduce the performance of any link. It does not lose accuracy and adapts to IP-layer path dynamics. Theoretical analysis and simulation results, in terms of monitoring cost and querying overhead, are also discussed in this paper.

1 Introduction

An overlay network is a virtual network formed by a subset of nodes in the underlying layer and virtual links composed of one or more hops on the lower-layer links. Recent research has shown a promising future for using application-layer overlay network to introducing new applications and services, e.g. multicast [1], Quality of Service (QoS), resilient routing, peer-to-peer file sharing, all without disrupting the operation of the lower IP layer. To better support the many upcoming overlay applications, researchers have proposed a generic *overlay service network* (OSN) [2, 3]. An OSN implements common functionalities among application-specific overlays, e.g. overlay link performance monitoring, topology construction, overlay service composition, and provides them as services to applications. It can coordinate the activities of multiple overlays and reduce overhead by avoiding repeating common tasks.

Monitoring overlay link performance, e.g. delay or loss rate, is a common task required by many applications. An overlay network with n nodes might need to monitor n^2 links, with each monitoring job incurring its own overhead. For overlay networks with large number of nodes, a scalable solution is necessary. In fact, several existing random measurement works have already incurred significant amount of overhead to the Internet. Reducing the monitoring overhead while maintaining the measurement accuracy is a challenging task that remains to be fully addressed.

This paper proposes a scalable monitoring service overlay network (MONET), which aims to measure the performance of overlay links and provide timely results to any querier. The key idea is based on the proposed *X-Set* concept (Sec. 3) through which overlay nodes

F. Boavida et al. (Eds.): NETWORKING 2006, LNCS 3976, pp. 513–524, 2006.

can share measurement information and deduce the performance of some links without directly measuring them. MONET can effectively reduce the monitoring overhead and distribute the measurement load among overlay nodes, all without scarifying accuracy. It also adapts well to lower-layer dynamics like IP path changes.

The rest of this paper is organized as follows. Sec. 2 presents some related work. Sec. 3 discusses *X-set*, which is the foundation of the link selection algorithm used in MONET. Sec. 4 presents the framework and detailed operations of MONET. We present some analysis in Sec. 5, simulation studies in Sec. 6, and conclude in Sec. 7. Due to space limitations, the detailed proof and additional simulation results are referenced for interested readers [4].

2 Related Work

Internet measurement is an active research field. Extensive work has been dedicated to inferring per-link performance when limited information is available [5, 6]. In overlay networks, measurement is focused on the performance of overlay links instead of individual IP links. Several works have been done to infer the distance between two arbitrary end hosts [7, 8, 9]. However, their approaches are only applicable to estimate the approximate end-to-end distances (delay), which is different from our goal of providing accurate overlay link performance information.

Shavit et al. [10] use algebraic tools to compute the link distances that are not directly measured. Given some tracers and some direct path measurement results, the proposed method can infer the performance of some paths or path segments. However, they do not deal with the selection of directly monitored links, sharing monitoring results, or providing scalable monitoring service, which are necessary for overlay networks. Chen et al. [11] also use an algebraic method to show how to use minimal linearly independent k paths to represent the performance of all n^2 paths. Both methods and MONET try to exploit the IP-layer information for reducing the overlay link monitoring overhead. The approach in [11] uses a centralized approach to determine directly monitored overlay links. In contrast, MONET proposes a distributed approach and tradeoff overhead reduction for scalability to large overlay network. In addition, MONET can cope with dynamic IP-layer path changes and avoid single point failure.

Tang et al. also propose approaches to reduce the number of directly monitored overlay links and track all the performance of all possible overlay links [12]. It has a centralized version and a distributed version. Their approaches are based on the assumption that an overlay link performance is approximately similar to the performance of its sub-segments, which can not provide accurate monitoring results.

3 X-Set

Although an overlay node can measure the performance to any other nodes, it is possible for overlay nodes to share information and reduce measurement overhead if there is sufficient knowledge about the IP topology. Fig. 1 depicts a simple scenario in which node C is on the path of AB. Although there are 3 overlay links, it is sufficient to monitor the delay of any two of them and then deduce the other. For an *additive* metric

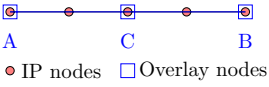


Fig. 1. On-Path Overlay Nodes

$$\overline{AB} = \overline{AC} + \overline{CB} \quad (1)$$

$$\widetilde{AB} = 1 - (1 - \widetilde{AC})(1 - \widetilde{CB}) \quad (2)$$

$$log(1 - \widetilde{AB}) = log(1 - \widetilde{AC}) + log(1 - \widetilde{CB}) \quad (3)$$

like delay, the relationship of the three links can be expressed with Eq. 1. Similarly, the loss rate of link AB ($\widetilde{AB}$), which is a *multiplicative* metric, can be found using Eq. 2. This paper focuses on additive metrics since a multiplicative metric can be transformed to additive metric at log-scale (Eq. 3).

For an overlay node A, if we map its paths to all other nodes onto the underlying IP topology, all the IP paths form a *source-based routing tree* (SRT) rooted as this node. Similarly, the IP paths from other nodes to A form a *destination-based routing tree* (DRT). Both source and destination-based routing trees can be decomposed into a set of basic components in the shape of a reverse "Y". We can use this basic component to reduce the number of overlay links we need to monitor. Consider a simple topology with four overlay nodes A, B, C and D. A and B need to monitor the performance of the overlay links (AC, AD, BC, and BD) connecting to nodes C and D. Based on the SRT of A or B, the two paths to C and D (from A to C, D and from B to C, D respectively) can be decomposed into two "Y"s. The different combinations of the two "Y"s are shown in Fig. 2, in which, X, Y and Z are non-overlay, on-path nodes. Note that Fig. 2 does not include every possible combination of two "Y"s. However, any other scenario can be reduced to one of the graphs in Fig. 2.

Equations in Table 1 present the relationship among the performance of overlay links for topologies illustrated in Fig. 2. For an additive metric, the performance of an

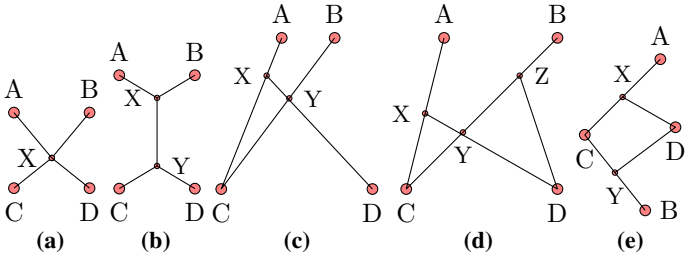


Fig. 2. Different Combinations of Two "Y"s

Table 1. Mathematical Expression of the Graphs in Fig. 2

Path	$\overline{AC}$	$\overline{AD}$	$\overline{BC}$	$\overline{BD}$
(a)	$\overline{AX} + \overline{XC}$	$\overline{AX} + \overline{XD}$	$\overline{BX} + \overline{XC}$	$\overline{BX} + \overline{XD}$
(b)	$\overline{AX} + \overline{XY} + \overline{YC}$	$\overline{AX} + \overline{XY} + \overline{YD}$	$\overline{BX} + \overline{XY} + \overline{YC}$	$\overline{BX} + \overline{XY} + \overline{YD}$
(c)	$\overline{AX} + \overline{XC}$	$\overline{AX} + \overline{XY} + \overline{YD}$	$\overline{BY} + \overline{YC}$	$\overline{BY} + \overline{YD}$
(d)	$\overline{AX} + \overline{XC}$	$\overline{AX} + \overline{XY} + \overline{YD}$	$\overline{BZ} + \overline{ZY} + \overline{YC}$	$\overline{BZ} + \overline{ZD}$
(e)	$\overline{AX} + \overline{XC}$	$\overline{AX} + \overline{XD}$	$\overline{BY} + \overline{YC}$	$\overline{BY} + \overline{YD}$

overlay link is the combination of the performance of all its sub-segments. Using a similar theory to that used by Chen et al. [11], if there are linearly dependent equations within a set of overlay link performance expressions, some of the overlay links can be removed from the set of links to directly measure without affecting the accuracy. The total number of overlay links these two nodes (A and B) need to directly monitor is the rank of this set of equations. The corresponding directly monitored overlay links are the linearly independent equations. For the equation sets in Fig. 2, it is easy to see that the equations in sets (a) and (b) can be decomposed into three equations, which means that the performance of the four overlay links in Fig. 2a and Fig. 2b can be obtained by directly monitoring three overlay links. For example, if node A monitors the performance of AD, node B monitors the performance of BC and BD, A can obtain the performance of AC since $\overline{AC} = \overline{AD} + \overline{BC} - \overline{BD}$.

Definition 1. ***X-Set****: For two overlay nodes, if their IP layer paths to the other two overlay nodes can be reduced to Fig. 2a or Fig. 2b, the four overlay links form an X-Set. The performance of all the four overlay links can be obtained by directly monitoring any three of them.*

The basic requirements for two *Y*s to compose an *X-Set* is that the two branching nodes of the two "Y"s overlap with each other such as node X in Fig. 2a and Y in Fig. 2b. Based on this, two nodes can cooperate with each other to find *X-Set*s (the details of which are described in the next section). As "Y"s are the basic components of SRTs, finding X-Sets is the basic method for two overlay nodes to cooperate and reduce the total number of directly monitored overlay links. More complicated combinations of the SRTs of two nodes can be partitioned into multiple X-Sets, which then allows the total number of directly monitored overlay links to be reduced.

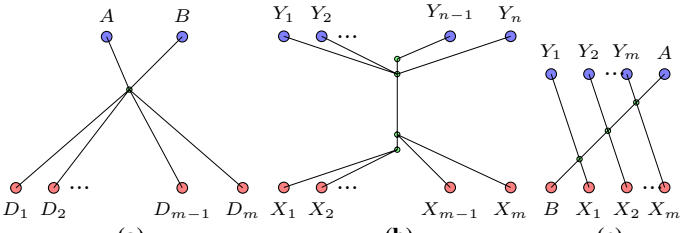


Fig. 3. Combinations of X-Sets

For example, in Fig. 3a, overlay node A and B need to track the performance of the $2m$ incident overlay links to the destination nodes (from D_1 to D_m). The graph can be seen as the combination of $m-1$ X-Sets ($\{A, B, D_1, D_2\}, \{A, B, D_1, D_3\}, \cdots \{A, B, D_1, D_m\}$). For the first X-Set, we only need to directly monitor three overlay links. For each of the remaining $m-2$ X-Sets, one only needs to directly monitor one additional link to obtain the performance of two links (as we already have the performance of AD_1 and BD_1). The total number of directly monitored links is $m+1$ instead of $2m$.

4 A Framework of Monitoring Service Overlay Network (MONET)

MONET assumes that an overlay link observes many more performance (delay or loss rate) changes than underlying IP path. An overlay node can identify its IP routing information to other nodes, by either querying other service modules or by *traceroute*. In MONET, overlay nodes independently determine their incident overlay links and continuously monitors their performance. They also share these monitoring results with a set of neighbors.

Given an IP topology $G(V, E)$ and a set of overlay nodes $V' \in V$, each overlay node independently chooses its set of directly monitored overlay links either based on its local information or by collaborating with other overlay nodes. These overlay links form the topology $G'(V', E')$ of the MONET, in which each link in G' is an IP path in G. Based on the MONET topology, each overlay node continuously monitors the performance of its incident overlay links. The links in the MONET topology are *directly monitored*. Other links in the corresponding full-mesh topology are called *indirectly monitored overlay links*, whose performance can be derived by the directly monitored results. In other words, MONET aims to track the performance of n^2 overlay links but incurs the least amount of monitoring overhead. Meantime, MONET aims to minimize the communication overhead and balance the load among the overlay nodes.

4.1 How Does MONET Work?

In MONET, each overlay node maintains an *overlay monitoring table* (OMT), which is an essential component to provide overlay link monitoring services. One entry is created for each adjacent overlay node in the full mesh topology. Each OMT entry has three fields: $DestID, MonitorBool, MethodList$. $DestID$ is the address of the neighbor – the destination of this overlay link. $MonitorBool$ determines whether the current overlay node (the OMT owner) should directly monitor the performance of the corresponding overlay link or not. If not, the $MethodList$ field includes the list of methods to obtain the performance of this link and the maximal query hops for each of these methods. For each indirectly monitored overlay link, an overlay node may have more than one method to obtain the corresponding overlay link performance. In our simulation studies (Sec. 6), we assume that each node only maintains one method for each indirectly monitored overlay link.

For example, in Fig. 1 and Fig. 2a, A can use one of the two methods to obtain the performance of link AC: $\overline{AB} - \overline{BC}$ or $\overline{AD} + \overline{BC} - \overline{BD}$. A needs to query one hop for the performance of BC. When a query arrives at node A for link AC performance, A first locates the corresponding entry for AC from its OMT. If the entry's $MonitorBool$ is true, it can directly return the overlay link performance. Otherwise, it will obtain the methods from MethodList and try each of them to obtain the link performance. Based on method $\overline{AB} + \overline{BC}$, besides checking the entry for AB, node A also needs to send a query to B for the performance of link BC, or, based on $\overline{AD} + \overline{BC} - \overline{BD}$, it will send a query for $\overline{BC}$ and $\overline{BD}$. If any performance query request returns, A can obtain the performance of overlay link AC. It is easy to see that a link performance query may take several query hops to return the performance. To balance the tradeoff between the

query distance and query overhead, a node can try each of the methods (or a subset of them) in parallel or sequentially.

4.2 Find Directly Monitored Overlay Links

Besides an OMT, each overlay node also needs to maintain two other data structures: a list of *Friend Nodes* and the corresponding list of "Y"s for each Friend Node. To fill each OMT entry, the overlay nodes can take the following two steps.

Algorithm 1. Finding "Y"s

$Y - Set \leftarrow \emptyset$ *//Initialization*
for each overlay node X **do**
  Retrieve the paths to every other nodes
  Construct the SRT rooted at X
  for each overlay nodes pair A, B **do**
    Find the branching node BN_{AB} *// the furthest common node of XA and XB in the SRT*
    if $BN_{AB} \neq X$ **do**
      Append $< A, B, BN_{AB} >$ into Y-Set

Algorithm 2. Load Balancing

Input: X-Set (A, B, X, Y), $S \leftarrow$ size of overlay
Require: ID_A<ID_B and $ID_X < ID_Y$
if $ID_Y < S * 1/(2^{1/2})$
  if $ID_B < S * 1/(2^{1/2})$ [*case 1*]
    $A \rightarrow \{AY, AX\}$, $B \rightarrow \{BX\}$
  else $A \rightarrow \{AX\}$, $B \rightarrow \{BY, BX\}$ [*case 2*]
else
  if $ID_B < S * 1/(2^{1/2})$ [case 3]
    $A \rightarrow \{AY, AX\}$, $B \rightarrow \{BY\}$
  else $A \rightarrow \{AY\}$, $B \rightarrow \{BY, BX\}$ [*case 4*]

First, each node independently identifies its list of "Y"s using Algo. 1. The main idea of Algo. 1 is to construct the SRT so that a node can locate the branching nodes and "Y"s. Based on the IP paths to other overlay nodes, a node can also find the possible scenarios as described in Eq.1. In addition, a node can also choose a set of overlay nodes as *friend nodes*, with which the node will share monitoring results. The selection of friend nodes is based on the IP path distance because the closer the two nodes are, the higher the chance that their incident overlay links will compose X-Sets.

In the second step, an overlay node will exchange its list of "Y"s information with the selected Friend Nodes. Based on the "Y" information from its friend nodes, a node can identify the *X-Sets* by comparing the common branching nodes of the two "Y"s for any two destination nodes. To balance the overhead from directly monitoring among the overlay nodes and to avoid the complicated negotiation procedure, an overlay node uses Algo. 2 to select its directly monitored overlay links.

The input to Algo. 2 is an X-Set with source nodes as A, B and destination nodes as C, D. The main idea is based on the ID values of the four nodes, which is a unique number defined by either IP address or MAC address. The probability of $(ID_X < ID_Y)$, $(ID_Y < S * 1/(2^{1/2}))$, $(ID_A < ID_B)$ and $(ID_B < S * 1/(2^{1/2}))$ are all $1/2$. Considering both case 1 and case 3, for the probability of $1/2$, node A only needs to monitor one overlay link for an X-Set. We can conclude that this algorithm balances the monitoring overhead (both sending and receiving measurement probing traffic) among the different overlay nodes without complicated negotiation procedures.

4.3 Dealing with Dynamic Network Condition

In MONET, each node periodically (much less frequently than link performance probing) performs traceroute or other methods to obtain the IP-layer path information. If a

node realizes that an IP-layer path to the other node changes, it will check whether there is any change in its set of "Y"s. If necessary, it will update some overlay links' monitoring methods. In addition, it will also send the "Y" update information to its friend nodes, which in turn may need to update their OMTs. The procedure of updating OMT entries is similar as adding OMT entries as discussed above.

Similarly, when an overlay node joins an existing overlay network, it first retrieves the IP-path information to other overlay nodes and chooses its friend nodes. After finding its "Y"s in its SRT and receiving "Y" information from its friend nodes, it can begin to find "X-Set" and fill its OMT entries one-by-one. If an overlay node needs to update its friend nodes set, it can also take similar steps.

In some cases, overlay nodes may not be able to retrieve the complete IP path information. For example, an IP path traceroute result could be like "69.110.237.117, *, 171.66.1.17, 171.67.255.249, *, *, 171.66.7.234". As the *X-Set* technique in MONET is based on the overlapping of two "Y"s' branching nodes, the incomplete path information will only affect the number of "Y"s each overlay node can find. It may result in the increase in the number of directly monitored overlay links. However, it will not affect the correctness and normal operations of MONET.

In summary, each MONET node independently (by exchanging information with a selected set of friend nodes) chooses which methods are used to track overlay link performance by either direct monitoring or indirect monitoring. Using the proposed techniques, MONET can effectively reduce the number (cost) of directly monitored overlay links without affecting the monitoring accuracy. In addition, it can quickly handle IP topology or IP path changes and dynamic overlay network membership.

5 Performance Analysis

5.1 Number of Overlay Links in MONET Topology

Fig. 3b and Fig. 3c shows the two different combinations of *X-Set*s. In Fig. 3b, node X_1 needs to monitor the performance of links from itself to node $Y_1, \cdots, Y_n$. Suppose X_1 realizes that the path of other $m-1$ nodes $(X_1, X_2, \cdots, X_m)$ to $Y_1, Y_2, \cdots, Y_n$ compose multiple X-Sets as shown in the graph. We can estimate the average number of links a node X_1 need to directly monitor in order to obtain the performance of all its links $(X_1Y_1, X_1Y_2, \cdots, X_1Y_n)$ based on the following analysis. First, X_1, X_2, Y_1 and Y_2 compose the first X-Set; the total number of directly monitored overlay links (for both X_1 and X_2) is 3. After this, if X_1 and X_2 want to monitor the links to one additional node (such as the links to Y_3, X_1Y_3 and X_2Y_3), they only need to directly monitor one more link to obtain the performance of two (X_1Y_3 or X_2Y_3). If another node (such as X_3) wants to monitor the performance to Y_1 and Y_2 (X_3Y_1 and X_3Y_2), it only needs to directly monitor one additional link (X_3Y_1 or X_3Y_2). Consequently, in order for all the nodes $(X_1, X_2, \cdots, X_m)$ to obtain the overlay links' performance to all the destination nodes $(Y_1, Y_2, \cdots, Y_n)$, the total number links $X_1 \cdots X_m$ need to directly monitor is $m+n-1$. On average, for each overlay link, one node needs to have $\frac{1}{m} + \frac{1}{n} - \frac{1}{mn}$ incident links in the MONET topology (average per node and per link directly monitoring cost) to obtain the performance of all the links.

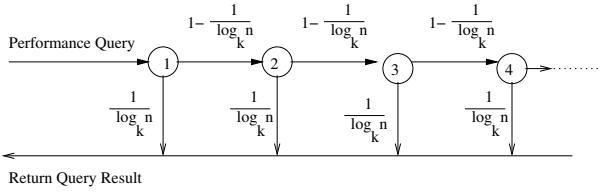


Fig. 4. Link Performance Query Processing Steps

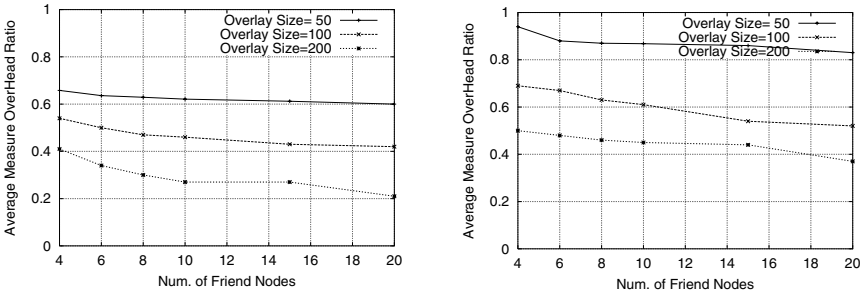
As mentioned above, the routing path of an overlay node to other nodes can be mapped to a SRT rooted at itself. For a connected graph, all the overlay nodes are located at the leaf nodes of other nodes' SRTs. Assume the total of n nodes are in the MONET and the average branching degree in the routing tree is k. The average height of the tree is h ($h = \log_k^n$). For a routing tree, it has different levels of sub-trees: the level 0 sub-tree is itself; level 1 sub-trees are the sub-trees that rooted at the children nodes of the root;...;level h sub-trees are the leaf nodes.

5.2 Overlay Link Performance Query Hops

A node of MONET does not monitor all adjacent overlay links directly. Therefore, it may need to query other nodes, which may repeat the similar process, to infer the performance of the link under request. We use *Link Performance Query Hops* to evaluate the average query distance to fulfill each overlay link performance query. As depicted in Fig. 4, the average directly monitoring cost of each overlay link is $\frac{1}{\log_k^n}$. The number of incident overlay links for each node is $n * \frac{1}{\log_k^n}$. A link performance query processing procedure is shown in Fig. 4. Suppose a query arrives at node 1. Node 1 has a probability of $\frac{1}{\log_k^n}$ to respond to the query without querying others. Otherwise, it will forward the request to the next node (e.g. node 2) based on its OMT. Node 2 will then repeat the same procedure: either returns the query result to node 1 with probability of $\frac{1}{\log_k^n}$ or sends another query based on its OMT to node 3. Consequently, the average query hops can be found as $\log_k^n - 1$. One can show that the upper bound of the directly monitoring cost for each overlay link is $\frac{1}{\log_k^n} - \frac{k}{\log_k^n * n}$. The cost is inversely proportional to the average height of the SRTs. Given a fixed number of overlay nodes, the smaller value of the average degree is in the routing tree, the lower monitoring cost each overlay link incurs (less directly monitored overlay links in the MONET topology).

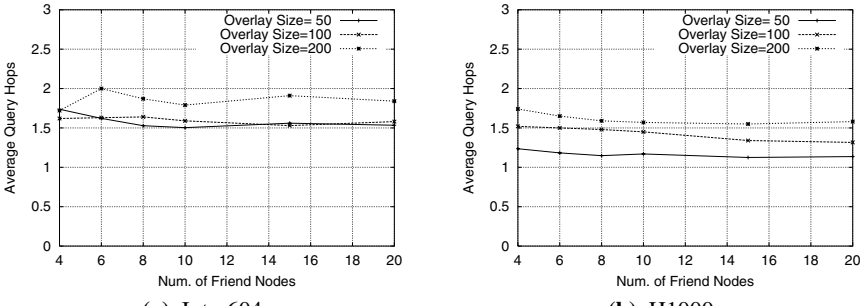
6 Simulation Study and Discussions

We evaluate the performance of MONET through simulation. The simulations are based on a real ISP intra-domain topology (Intra604) taken from Rocketfuel [13] and three topologies generated by BRITE [14]. *Intra604* has 604 nodes, 4547 directed links and an average node degree of 7.5. For the other three topologies, *W1000* is a router-level Waxman [15] topology with 1000 nodes. The other two (*H1000* and *H5000*) are two 2-layer hierarchical topologies with the lower level based on Waxman model and the higher level based on Barabasi-Albert model, with 1000 and 5000 nodes respectively.



(a). Intra604 **(b).** H1000

Fig. 5. Monitoring Overhead vs. Num. of Friend Nodes



(a). Intra604 **(b).** H1000

Fig. 6. Average Indirectly-Monitored Overlay Link Performance Query Hops

For the topologies generated by BRITE, each node is adjacent on average of two undirected links, which leads to an average node degree of 4.0. The IP layer use shortest path based routing. We focus on the following performance metrics: average query hops for indirect monitored overlay links, average overlay link monitoring overhead, monitoring overhead balancing results, and OMT table updates under dynamic IP-layer change. Due to space limitation, we only present the results on *Intra604* and *H1000*.

6.1 Monitoring Overhead

We use *Monitoring Overhead Ratio* (MOR) to evaluate the performance of MONET in reducing the monitoring overhead of each node to provide constant link performance monitoring service. For an overlay node, MOR is defined as:

$$MOR = \frac{\text{\# of Adjacent Directly Monitored Overlay Links}}{\text{Overlay Network Size}} \tag{4}$$

The directly monitored overlay link means that the overlay node keeps sending probing traffic to monitor the overlay link performance. It is easy to see that the smaller value of MOR means that MONET can provide better performance in terms of decreasing the monitoring overhead. Fig. 5 shows that the average MOR for overlay networks of various sizes and numbers of friend nodes on top of the different IP topologies. From the

simulation results, one can observe that the increase in the number of friend nodes will reduce the average MOR, which means the monitoring overhead will decrease. This is because each node has higher chance to find X-Sets with its neighbors. However, as we mentioned above, the "Y" information needs to be shared between friend nodes. The larger number of friend nodes means that higher amounts traffic will be exchanged during dynamic IP-layer path changes. When considering different sizes of overlay networks on the same underlying IP topology, we can observe that the performance of MONET varies greatly. The larger an overlay network is, the less the average MOR is. This is because each node can have more candidate nodes to choose as friend nodes. This will result in more X-Sets, which means that the number of directly monitored overlay links can potentially decrease.

6.2 Average Query Hops

An overlay node needs to query others when a query for an indirectly monitored overlay link arrives and then relay the answer back. The number of query hops determines the response delay and the accuracy of the performance value. Fig. 6 shows that the average number of query hops for all indirectly monitored links. We can observe that the average query hops are all below 2.0 for the various simulated scenarios. The average query hops in Fig. 6a (between 1.5 and 2.0) is higher than Fig. 6b (between 1.0 and 1.5). This is because the first topology is smaller, resulting in a higher probability of finding X-Sets. Considering together with the simulation results of the average MOR and average query hops, we can conclude that lower MOR is correlated to longer query hops which agrees with our previous analysis results. In addition, we can observe that increasing the number of friend nodes only slightly affects the average query hops.

6.3 Balancing the Monitoring Overhead

MONET aims to balance the overlay link monitoring overhead among all the overlay nodes as described in Algo. 2. For comparison, we consider another none-load-balancing monitoring overhead distribution method: if A and B (assume $ID_A < ID_B$) find the overlay links connecting to X and Y (assume $ID_X < ID_Y$) form an *X-Set*, A will never monitor link AX but always deduce its performance based on the values of the other three links. Note that both methods allows overlay nodes to share the monitoring overhead and collaborate without complicated negotiation and message exchanges.

Fig. 7 presents the effect of load balancing based on *Intra604* with 4 friend nodes. The x-axis values are the different bins of MOR values while the y-axis shows the numbers of overlay nodes within the corresponding bins. With load balancing, more nodes are located in the bins whose value are closer to the average MOR. This suggests that Algo. 2 can effectively distribute the overhead among nodes. In contrast, if load balancing is not available, some nodes will have high monitoring overhead while others are lightly loaded.

6.4 Effect of IP-Layer Path Change

The operation of MONET is based on the IP-layer path information. If there is an IP-layer path change in the overlay links, some X-Sets will be added or deleted, which

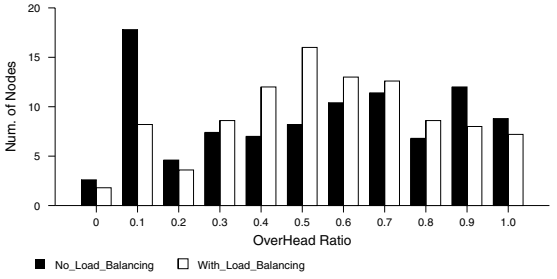


Fig. 7. Distribution of Monitoring Overhead Ratio

Table 2. The Effect of IP-layer Path Changes (Overlays on Top of Intra604 Topology)

Overlay Size	# of Friends	IP-layer path Failure Ratio	# of IP Link Failures	# of Overlay Path Changes	# of "Y" Changes	# of OMT Updates
50	4	0.001	6	14.66	3.75	5.6
50	8	0.001	6	15.64	4.70	3.43
50	10	0.001	6	13.54	5.21	3.45
50	15	0.001	6	10.18	4.82	5.30
50	4	0.002	11	24.76	6.86	4.48
50	8	0.002	11	23.68	7.70	5.05
50	10	0.002	11	25.30	9.5	7.0
50	15	0.002	11	31.90	8.6	10.03
100	4	0.001	6	47.75	9.2	6.14
100	8	0.001	6	45.10	9.58	20.35
100	10	0.001	6	50.0	10.38	29.4
100	15	0.001	6	51.38	13.3	40.92
100	4	0.002	11	122.76	18.5	14.96
100	8	0.002	11	113.79	19.12	25.09
100	10	0.002	11	124.63	27.13	54.43
100	15	0.002	11	125.00	22.73	70.13

leads to the updates of OMT table. In this paper, we use *Intra604* as an example to investigate the effect of IP-layer path changes. We first randomly form an overlay network with size 50 or 100. After this, each node runs the MONET to set up its OMT table. After the system stabilizes, we randomly fail some IP-layer links without losing the IP-layer connectivity. After this, the affected OMT entries will the refilled by MONET. The relationships between IP-layer, overlay layer, number of "Y" changes as well as the OMT table updates are shown in Table 2. From the results, we can observe that the increase of the IP-layer link failure ratio will increase the number of changes in the overlay links' IP-layer paths. This is because that whenever the average number of friend nodes is increased, the affected number of X-Sets (added or deleted) also will be increased. This will result in larger number of OMT table updates. The OMT updates include the changes between direct overlay link monitoring and indirect monitoring, as well as the changes between different indirect monitoring methods. However, even under higher IP-layer path failure ratios (0.001 or 0.002), the average number of OMT

updates is very small, less than 0.5 entry per node. This is because that even if there are a lot of overlay link IP-layer paths change, the new paths will most likely take similar paths to bypass the failed links. Consequently, even though the locations of X-Set branching nodes change, the composition and the number of X-Sets will more or less remain stable.

7 Conclusion

This paper proposed a framework called MONET to efficiently monitor and provide accurate overlay link performance information. The important mission of MONET is to reduce the monitoring cost while maintaining monitoring accuracy. MONET uses a distributed approach that can evenly distribute the path monitoring overhead and easily deal with IP-layer path changes. We also presented some analysis and simulation results in terms of monitoring overhead reduction, link performance query hops and monitoring load balancing.

References

1. Chu, Y.H., Rao, S.G., Zhang, H.: A case for end system multicast. In: Measurement and Modeling of Computer Systems. (2000)
2. Lakshminarayanan, K., Stoica, I., Shenker, S.: Building a flexible and efficient routing infrastructure: Need and challenges. Technical report, UC Berkeley UCB/CSD-03-1254 (2003)
3. Braynard, R., Kostic, D., Rodriguez, A., Chase, J., Vahdat, A.: Opus: an overlay peer utility service. In: IEEE OpenArch'02. (2002)
4. Li, Z., Yuan, L., Mohapatra, P.: An efficient overlay link performance monitoring technique. Technical Report CSE-2005-28, Computer Science, University of California, Davis (2005)
5. Padmanabhan, V.N., Qiu, L., Wang, H.J.: Server-based inference of interent performance. In: Proc. IEEE INFOCOM. (2003)
6. Caceres, R., Duffield, N., Horowitz, J., Towsley, D.: Multicast-based inference of network-internal loss characteristics. In: Proc. IEEE INFOCOM. (1998)
7. Ng, E., Zhang, H.: Predicting internet network distance with coordiantes-based approaches. In: IEEE INFOCOM. (2002)
8. Tang, L., Crovella, M.: Virtual landmarks for the Internet. In: ACM SIGCOMM/USENIX IMC. (2003)
9. Dabek, F., Cox, R., Kaahoek, F., Morris, R.: Vivaldi: A decentralized network coordinate system. In: ACM SIGCOMM. (2004)
10. Shavitt, Y., Sun, X., Wool, A., Yener, B.: Computing the unmeasured: An algebraic approach to internet mapping. In: Proc. IEEE INFOCOM. (2001)
11. Chen, Y., Bindel, D., Katz, R.H.: An algebraic approach to practical and scalable overlay network monitoring. In: ACM SIGCOMM. (2004)
12. Tang, C., McKinley, P.K.: On the cost-quality tradeoff in topology-aware overlay path probing. In: ICNP. (2003)
13. Spring, N., Mahajan, R., Wetherall, D.: Measuring isp topologies with rocketfuel. In: Proc. ACM SIGCOMM. (2002)
14. Medina, A., Lakhina, A., Matta, I., Byers, J.: BRITE. `http://www.cs.bu.edu/brite/`(2002)
15. Waxman, B.M.: Routing of Multipoint Connections. IEEE JSAC (1988)