

A Distributed Approach to Topology-Aware Overlay Path Monitoring

Chiping Tang and Philip K. McKinley

Software Engineering and Network Systems Laboratory
Department of Computer Science and Engineering
Michigan State University
East Lansing, Michigan 48824
{tangchip,mckinley}@cse.msu.edu

Abstract

Path probing is essential to maintaining an efficient overlay network topology. However, the cost of complete probing can be as high as $O(n^2)$, which is prohibitive in large-scale overlay networks. Recently we proposed a method that trades probing overhead for inference accuracy in sparse networks such as the Internet. The method uses physical path information to infer path quality for all of the $n \times (n - 1)$ overlay paths, while actually probing only a subset of the paths. In this paper we propose and evaluate a distributed approach to implementing this method. We describe a minimum diameter, link-stress bounded overlay spanning tree, which is used to collect and disseminate path quality information. All nodes in the tree collaborate to infer the quality of all paths. Simulation results show this approach can achieve a high-level of inference accuracy while reducing probing overhead and balancing link stress on the spanning tree.

1. Introduction

Overlay networks, in which end hosts form a virtual network atop a physical network, can provide an adaptable and responsive chassis on which to implement communication services for distributed applications. Overlay networks have been used to support end-system multicast, structured peer-to-peer systems, global event notification services, resilient routing, and denial-of-service attack prevention. Each logical link in the overlay network comprises an end-to-end path in the physical network. The choice of paths to be included in the overlay network may have significant effect on performance. In dynamic environments such as the Internet, the quality (in terms of loss rate, available bandwidth, delay, etc.) of paths between nodes may change frequently. Hence, it is important for overlay nodes to monitor the quality of paths and adjust the overlay topology accordingly.

To monitor a path, a node can periodically send probe packets to the node at the other end of the path, which returns acknowledgement packets. From the delay characteristics and delivery status of these probe/acknowledgement packet pairs, a node can infer the quality of the path. In complete pairwise probing, as employed in RON [2], each node probes the paths from itself to all other nodes. Although this approach can produce complete and accurate quality results, the probing overhead is quadratic in the overlay network size and can lead to high link stress even in medium-sized overlay networks of 100 nodes [18].

We recently proposed an approach to reduce probing overhead by exploiting information of the underlying physical network topology [18]. This method is based on two observations. First, the quality of a path can be inferred from the quality of its constituent subpaths, or *segments*. Second, in a sparse network such as the Internet, the paths in an overlay network overlap considerably. Hence, by probing only a subset of the $n \times (n - 1)$ paths, we can infer quality information on all segments in the overlay network, from which we can produce bounded approximations on the quality of all the paths. In [18], we described a centralized implementation strategy, in which a leader is elected to coordinate the probing and inference process. This strategy is well-suited to overlay networks in which management decisions are already centralized. However, in systems where the nature of the overlay network is distributed, the leader is a potential performance bottleneck and a single point of failure. In addition, the stress on the links close to the leader may be high. Moreover, overlay nodes in systems such as RON may require global path quality information to make routing decisions locally. Having the leader broadcast the quality information of all the $n \times (n - 1)$ paths is not practical and counter to the objective to reduce overhead.

In this paper we propose a distributed implementation approach designed to address these issues. We use a minimum diameter, link-stress bounded overlay spanning tree to disseminate path quality information. Special steps are

taken to minimize the bandwidth consumption. All nodes participate in the monitoring process by executing the same algorithm. At the end of each probing round, every node has acquired all the path quality information. Simulation results show this approach can achieve a high-level of inference accuracy while reducing probing overhead and balancing link stress on the spanning tree.

The remainder of the paper is organized as follows. In Section 2, we describe related work in overlay network monitoring and network management. In Section 3, we review our inference and path selection algorithms. We introduce the general architecture of our monitoring system in Section 4 and describe several improvement techniques in Section 5. We present the performance evaluation results in Section 6, and make concluding remarks in Section 7. Due to space limitations, some details are omitted here, but may be found in [17].

2. Background and Related Work

Since the performance of overlay networks is sensitive to changes in path quality, path probing is essential for maintaining efficient overlay networks. Systems targeting small overlay networks usually employ pairwise probing. Larger overlay networks employ techniques such as hierarchy, approximation, or aggregation to improve the probing scalability. For example, application-level multicast protocols can take advantage of the tree structure to reduce probing overhead. On the other hand, peer-to-peer systems use randomization to choose neighbors for each member, and some works address how to generate a good random node subset for probing. Our approach to reducing probing overhead is orthogonal in that we focus on the tradeoff between probing cost and probing accuracy, instead of on the tradeoff between probing cost and probing completeness.

One disadvantage of overlay networks is communication inefficiency caused by discrepancies between the logical and physical topology (e.g., overlay paths may be longer and overlap more than necessary). Although physical topology information is usually not available at end nodes, many end-to-end approaches [5, 19] can be used to obtain, or infer, such information. Many overlay systems can benefit from physical topology information, and increasingly protocols are designed to exploit such information. In [11], Kwon and Fahmy propose a protocol that builds efficient multicast trees at the application level based on the topology information. In [7], Cui, Stoica and Katz propose a correlated link failure probability model to allocate backup paths in overlay networks. Their approach implicitly exploits the physical network topology information to build the failure probability model. The methods we propose in [18] explore how overlay network probing might benefit if topology information is widely available.

Our method of selected probing and path quality estimation is similar to network tomography approaches [3, 6] that infer link delay, loss rate, or available bandwidth from end-to-end measurements. However, those methods primarily address quality inference for *physical* links, while our interest is in end-to-end *path* quality. In addition, since network tomography methods are designed to support network management, they are usually executed offline and can therefore be computationally intensive. On the other hand, an overlay monitoring system needs to be lightweight and executed online. The distributed approach proposed in this paper facilitates the integration of our tomography mechanisms [18] into a wider class of overlay network systems.

Since the proposed method is distributed, overlay nodes periodically exchange information with one another. To mitigate this overhead, we construct link-stress-aware spanning trees to distribute the communication load. Although load balancing is a well-studied problem, to our knowledge the problem of link-stress balancing in overlay networks has not been addressed previously. We propose a simple algorithm to construct a minimum diameter, link-stress bounded overlay spanning tree. The algorithm is based on the BCT algorithm [15] for the limited diameter, residual-balanced (LDRB) spanning tree problem. Although we developed the new algorithm specifically for overlay network monitoring, it may be of more general use. Finally, we note that spanning trees have been used previously in the context of routing algorithms as a way to reduce the cost of all-to-all flooding [10]. Although our motives are similar, in that we want to reduce the cost of all-to-all probing, we exploit the relationship between our inference algorithm and the tree structure to reduce the bandwidth consumption.

3. Inference and Path Selection

3.1. Notations

An overlay network is a logical abstraction of the underlying physical network. Figure 1 shows the composition of an overlay network, with the bottom layer representing the physical network, and the top layer representing the overlay network. Formally, the overlay network is represented as a graph $G = \langle V, E \rangle$, where the vertex set V is the collection of all overlay nodes, and the edge set E is a collection of paths between overlay node pairs. All routers, as well as end nodes not involved in the overlay network, are abstracted away from the vertex set V . Similarly, the physical links of the selected overlay paths are not included in the edge set E .

In sparse networks, overlay paths often overlap. As shown in Figure 1, path $AB = (AE, EF, FB)$ shares the physical links AE and EF with paths $AC = (AE, EF, FG, GH, HC)$ and $AD =$

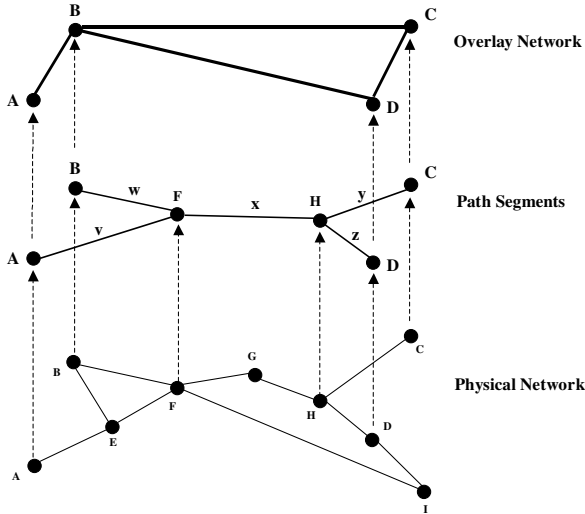


Figure 1. An overlay network, showing path segments and physical network.

(AE, EF, FG, GH, HD). To represent this property, we extend the overlay network model defined above to include path segments. A path segment is a subpath of a physical path and comprises one or more physical links. Every overlay path in turn comprises one or more path segments. In relation to the paths in an overlay network, path segments are defined as follows:

Definition 1: A path segment is one of the maximal subpaths in a path such that all the inner vertices on the subpath are not incident to any other physical links in the overlay network.

For a given overlay network, we construct the path segment set S as follows. For each overlay path, find its corresponding physical path. If the path overlaps with any segment in S , split the path into several subpaths so that the subpaths are either identical to one segment in S , or they are disjoint with all segments in S . If this is infeasible because a segment in S contains some of the subpaths but is not identical to them, then split those segments into subpaths. Repeat this process until any two subpaths or segment-subpath pair are either disjoint or identical. Add the newly generated subpaths to S as new segments. Discard any redundant segments. If a physical path does not overlap with any segment in S , add it to S as a single segment. Select the next overlay path and repeat this process.

After the construction of path segment set S , every overlay path can be expressed as one or several path segments in S . The construction algorithm guarantees that each path segment in S is disjoint with all other segments. Essentially this process changes a set of overlay paths to a set of path segments. The middle layer in Figure 1 shows the 5 path segments found in the overlay network.

3.2. Inference Algorithm

Our approach to quality estimation is based on the following assumptions. First, we assume paths between pairs of overlay nodes significantly overlap. Experiments on real Internet topologies, where the average vertex degree is a constant [9], reveal that the number of segments in an overlay network is much smaller than the number of paths [18]. Indeed, the number of segments in a sparse network is usually $O(n)$ or $O(n \log n)$, depending on the topology, where n is the number of overlay nodes. Our inference algorithm exploits this property to infer the quality of all path segments, from which it further calculates the quality of all the paths. Second, we assume route changes are much less frequent than path quality changes. This property is generally true since a route change is usually caused by path quality change while the reverse does not necessarily hold. Experimental results also show that Internet paths are relatively stable [20]. Third, we assume packets traversing the same physical link within a short time period will experience similar channel conditions. Many other end-to-end measurement applications employ algorithms based on this assumption [8]. Fourth, we assume the physical link composition of every path is known by at least one overlay node. Although this assumption has not generally held in the past, increasingly end node techniques and tools such as *traceroute*, *topology servers* [14], and *network tomography* [6] can be used to obtain physical topology information. Such information is also used in other overlay network projects [7, 11].

The inference algorithm we use in this study, called the *minimax inference algorithm*, is one that we proposed in [18]. The algorithm is applicable to metrics such as packet loss status and available bandwidth, and is based on the following observations. For such metrics, the quality of a given segment is bounded below by the maximum quality of those paths that are probed and contain the segment. Moreover, the quality of an unprobed path in the overlay network is bounded above by the minimum quality among its constituent segments.

Let us demonstrate the behavior of the minimax inference algorithm through an example using the network in Figure 1; complete details of the algorithm can be found in [18]. Assume node A probes nodes B and C simultaneously, while node C probes node D at roughly the same time. Assume further nodes A and C receive the corresponding acknowledgements from nodes B and D respectively, but that the acknowledgement from node C to A does not arrive. From these results we can infer that either the probe packet or the acknowledgement packet between A and C is lost. The loss could have occurred on any of segments v , x , or y (or at one of the incident vertices, due to a queue overflow). Assume the segment loss status is static within a short time interval, *i.e.*, either all packets traversing

a segment within this period are lost, or all of them are loss-free. Under this assumption, since nodes A and C have received acknowledgements from B and D , respectively, we can assume that segments v , w , y , and z are loss-free. Therefore segment x must be in a loss state. Moreover, from these results we can infer that the other paths containing segment x (AD , BC , and BD) are also in a loss state, without actually probing them. Similarly, we can infer lower bounds on available bandwidth on unprobed paths.

3.3. Path Selection Algorithm

We emphasize that the minimax algorithm is an approximation algorithm, enabling us only to infer lower bounds on path quality. The accuracy of the inference, or the difference between the actual quality and the inferred lower bound, depends on the distribution of the quality values and the selection of the paths to be probed. For example, if all the segments in Figure 1 except v were loss-free, and we happened to select paths AB , AC , and AD for probing, then the probing results of all three paths would indicate loss states. Moreover, the algorithm would also infer paths BC , BD , and CD to be in a loss state, even though they are actually loss-free. However, as more paths are probed, the lower bounds can be raised closer to the actual quality values.

We adopt a two-stage path selection algorithm. In the first stage, we select a minimum set of paths that covers all the path segments. This problem resembles the minimum weighted set cover problem [4], and we use a greedy algorithm to obtain an approximate solution. In the second stage, we continue to add paths to the set until the number of selected paths equals an application-specified threshold K . In this stage, we try to balance the stress, or the number of traversing paths, on each segment. We use a set-cover-like algorithm to add paths. The basic idea is to select the path that maximizes the number of segments for which the stress is made closer to the average.

3.4. Quality Estimation

In [18], we present results of an extensive study of path quality estimation. Those results show that, depending on the topology, the inference and path selection algorithms can provide quality estimation with up to 90% average accuracy for all paths with $O(n \log n)$ probing overhead. For example, Figure 2 shows the result for available bandwidth estimation with $O(n \log n)$ probe packets on the actual Internet AS-level topology as of February 2000 [12]. The first stage alone of the minimax algorithm (labeled All-Bounded) achieves over 80% average accuracy. Increasing to $n \log n$ probes raises the average accuracy to over 90%.

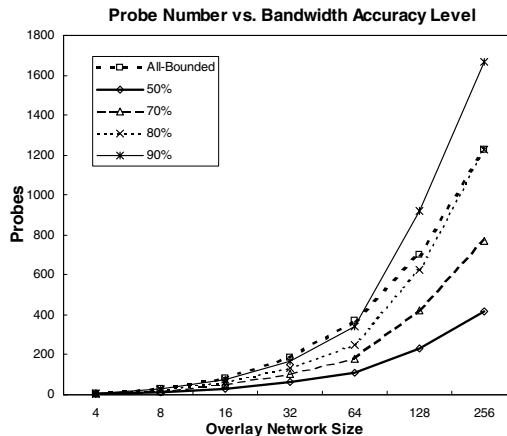


Figure 2. Number of probe packets vs. available bandwidth estimation accuracy [18]

4. System Design and Operation

In this section we describe a distributed approach to executing the minimax inference algorithm. Essentially, each node in the overlay network performs the tasks of probing and inference, using a spanning tree to exchange results with the other nodes. We refer to this tree as the *dissemination tree*. Here, we focus on the basic operation using a relatively simple minimum spanning tree. In the next section, we describe optimizations to reduce link stress and bandwidth consumption.

In order to limit the time required for a probing and inference calculations, it is desirable to constrain the diameter of the dissemination tree. A diameter constrained minimum spanning tree can be constructed using any of the proposed algorithms [1]. After the tree is constructed, the center of the tree is located using a simple algorithm: select an arbitrary node A and use depth first search to find the node, B , farthest away from A in the tree. Next, find node C farthest away from B . Then a center of path BC is also a center of the tree. We identify the center as the root of the spanning tree, and every node is assigned a *level* value denoting the distance to the root in terms of tree edges.

The system requires network topology information to be available at at least one node. We distinguish two cases: (1) all nodes maintain consistent topology and overlay membership information, and (2) some nodes do not have this information. In the first case, each node independently handles member joins and leaves, computes path segments, and identifies the set of paths it should probe using the path selection algorithm. Since the topology information is consistent at each node, and the path selection is deterministic, the path sets should be identical at all nodes. In the second case, a node with topology information is elected as a leader that handles member joins and leaves, generates segments, and computes the path set for each node. Unlike a central-

ized algorithm, the leader node does not execute the inference algorithm. Instead, it simply sends to each node the set of selected paths that are incident to that node, with the constituent segments of the paths specified.

The system operation is shown in Figure 3. In both cases, our system comprises a spanning tree rooted at its center, and each node has assigned to it a (possibly empty) set of incident paths that it should probe. The composition of these paths in terms of segments is also known at each node. We use an unreliable network protocol such as UDP to send probe/acknowledgement packets, and we use a reliable protocol such as TCP for communication along the tree edges.

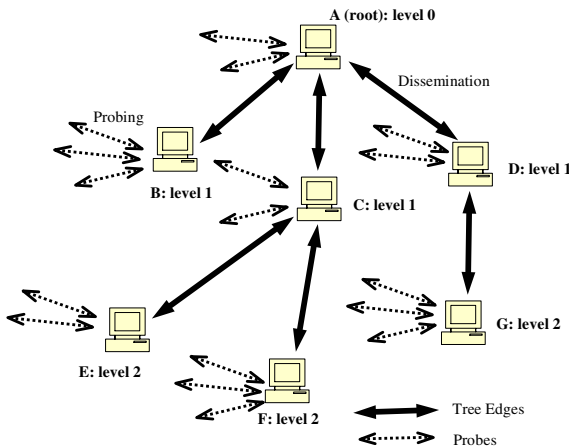


Figure 3. System operation.

In our system the quality inference and dissemination processes are integrated. Each node participates in the dissemination process in an up-down fashion by passing results up and down the spanning tree. When the dissemination terminates, each node has obtained the bounded quality estimation for all segments.

Any node in the system can start the procedure by sending a “start” packet to the root, which forwards the packet down to every node in the tree. On receiving the start packet, a node sets a timer according to its *level* value before conducting the probing. The purpose is to have all nodes start probing at approximately the same time so that they can experience the same quality on common segments.

When the timer fires, a node selects the paths incident to it from the probing set and sends probing packets to the nodes at the other end of the paths. It then derives path quality from the probing results and assigns the path quality as segment quality to all the segments in a selected path. Since each node only probes a small subset of paths, it needs to exchange the probing results with other nodes by disseminating its inferred quality of segments.

The dissemination starts at the leaf nodes. Each leaf node sends the quality of segments in its probed paths to its parent in the spanning tree. Upon receiving such a packet, a

node compares the quality of segments in the packet with the locally inferred segment quality. If the quality of a segment in the packet is lower than the locally inferred quality of the same segment, then this quality information in the packet is discarded, otherwise the local quality value is updated.

After the node has received and processed such packets from all of its children, it sends all the local inferences and inferences received from its children up to its parent. The parent will process the packet using the same method. The procedure repeats until the root of the tree is reached. The root then compares the inferences from different branches. For each segment, the maximum inference is selected as the quality value of this segment. Then the root sends down along the spanning tree a packet containing all the inferred segment quality to all the nodes. When receiving such packets, a node updates its local inference by using the inferred quality in the packet. This round of probing terminates when all the leaf nodes have processed the packet.

The computational complexity of this approach is $O(|S|)$ at all nodes. Next, we derive the communication overhead in terms of packet number and bandwidth consumption. Let a be the size in bytes of the quality information of a single segment, including the segment ID and its quality value. Assume $a = 4$ in a typical system. In the uphill stage, each node sends exactly one packet to its parent, and the size of the packet depends on the level of the node. The largest packets are the ones sent to the root by its children. For example, in Figure 3, leaf node E probes two paths. Assume there are a total of 16 segments in these two paths. Then node E would send only 64 bytes to its parent node C . On the other hand, node C needs to send the information about 7 paths (2 from E , 3 from F , and 2 local) to its parent node A , including probably 50 or so segments that correspond to about 200 bytes. Assume the root has c children, then on average the packet size at this level is $a|S|/c$, since the root is expected to receive the information about all $|S|$ segments. In the downhill stage, the root sends quality information for segments to all other nodes along the spanning tree. Hence, the size of all such packets is $a|S|$. Overall, the total number of packets in one probing round (excluding probing packets) is $2n - 2$, *i.e.*, two times the number of tree edges. The bandwidth consumption on each path is $O(|S|)$. In the next section we will show how to reduce the bandwidth consumption.

5. Enhancements

In our approach we disseminate the segment quality information through the diameter constrained minimum spanning tree. Since the size of this information is moderate in small and medium scale overlay networks, we expected the bandwidth consumption on each link is also small. How-

ever, we noticed in our simulation that the bandwidth consumption on some links is relatively high, as shown in Figure 4. In this experiment, we randomly assign 64 overlay nodes to 6474 vertices in a real AS-level Internet topology (see Section 6 for description of the topology “as6474_64”), and construct a diameter constraint minimum spanning tree. Simulation results show that over 90% of the links have a stress no higher than 1, with bandwidth consumption below 1KB. However, some links have a stress value around 10, and one link even has a stress value of 61, which corresponds to a bandwidth consumption of about 300KB. This is due to the fact that multiple tree edges may share segments, thus the link stress on some segments may be much higher than in the common case. This is particularly problematic since the volume of information disseminated in our system may be large. Clearly we need to limit the worst-case bandwidth consumption on segments.

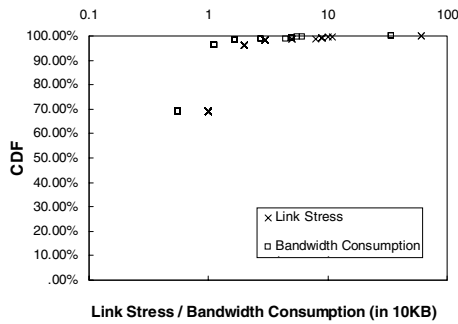


Figure 4. Unbalanced link stress and bandwidth consumption.

In this section we describe two enhancements to our basic system. First, we consider an alternative topologies for the dissemination tree that reduce link stress. We introduce the minimum diameter, link-stress bounded (MDLB) overlay spanning tree problem and propose an approximation algorithm to solve it. Next, we discuss how to use simple methods to reduce bandwidth consumption during the process of information exchange.

5.1. MDLB Problem

In our system we need to balance the node degree as well as the link stress so that there will be no potential performance bottleneck on nodes and communication links. We argue that modern network nodes usually have high computational power and thus can process a large number of incoming packets provided the communication links could afford to keeping the incoming rate. Therefore we focus more on balancing link stress than node degree. We suggest using a minimum diameter, link stress bounded overlay spanning tree to disseminate the quality information.

Definition 2: Minimum diameter, link stress bounded (MDLB) overlay spanning tree problem

Given an undirected graph $G = (V, E)$, and another undirected complete graph $G' = (V', E')$ denoting an overlay network on G , where $V' \subseteq V$, $E' = \{e' | e' \subseteq E \wedge IsPath(G, e')\}$, a cost $c(e) \in \mathbb{Z}^+$ for each $e \in E$, and a cost $c(e') \in \mathbb{Z}^+$, $c(e') = \sum_{e \in e'} c(e)$ for each $e' \in E'$, link stress $r(e) = |\{e' | e' \in E' \wedge e \in e'\}|$; find a spanning tree T of G' with minimum diameter, subject to the constraint that $r(e) \leq r_{max}(e)$, for all $e \in E$.

MDLB problem is similar to the minimum diameter, degree-bounded spanning tree problem (MDDB) [15]. In MDDB problem, the goal is to find a spanning tree that has a bounded maximum node degree with minimum diameter. This problem is NP-complete [15]. In [17], we show MDLB is also NP-complete by reducing MDDB to MDLB.

Theorem 1. The decision version of MDLB - finding a spanning tree with diameter bound B and a link stress constraint $r_{max}(e)$ for each link, is NP complete, for $0 \leq r_{max}(e) \leq |V'|(|V'| - 1)/2$.

Although the MDLB problem looks similar to the MDDB problem, they are fundamentally different. We cannot convert a MDLB problem to a MDDB one by simply constructing a graph with swapped vertex and edge sets. Actually in MDDB we only care about the node degree allocation. The actual path between any node pair is unimportant. Therefore the problem makes no much difference on overlay or physical networks. In MDLB, however, the actual path between a node pair will affect the maximum link stress value. For spanning trees in a physical network, the maximum link stress is always 1. Hence, the MDLB problem makes sense only on an overlay network. Figure 5 shows an example where a good solution to MDDB is not a valid solution to MDLB. In this example, the degree and link stress constraints are both set to 3. Consider the link stress resulting when the paths in the overlay network are mapped to physical links. This example shows the MDDB solution does not satisfy the link stress constraint, since the addition of the tree edge CG makes the stress on the bridge link be higher than 3.

We use a heuristic algorithm for the MDLB problem, which is similar to the BCT algorithm [15] for the MDDB related LDRB problem. In this algorithm, we build a spanning tree incrementally. Let T be the partial tree, $dis(u, v)$ be the distance between node u and v , and $diam(T, u)$ be the longest path in T starting from node u . At each step, we select a node $u \notin T$ such that $\min_{v \in T} d(u, v) + diam(T, v)$ is minimized while the stress on each link e (equivalently, its corresponding segment) is bounded by $r_{max}(e)$, and add it to the tree at v . The stress of the links in (u, v) is incremented by 1 after the addition.

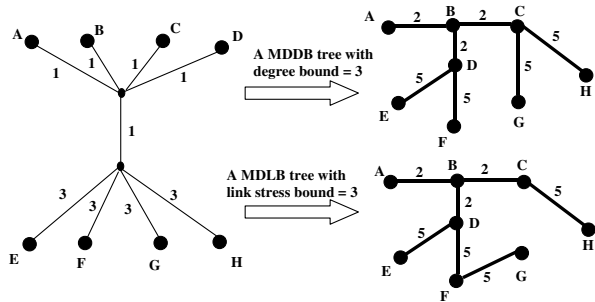


Figure 5. An example where the performance of MDDB and MDLB differ.

Sometimes it is difficult to find a tree that satisfies the link stress constraints while having a reasonably small diameter. If the resulting diameter is too large, we can relax the link stress constraint and repeat the algorithm. To increase the chance of obtaining a better result, we can interleave the MDLB algorithm with another tree algorithm. The algorithm finds a spanning tree with a bounded diameter and minimum link stress (BDML). In each step the algorithm finds a node such that one of its paths to the tree has the minimum link stress and satisfies the diameter constraint. This algorithm does not guarantee that the link stress of the tree edges is below a constraint value. We combine these two algorithms to find better spanning trees. First we initialize the diameter constraint value and the link stress constraint value. Then we run the BDML algorithm with this diameter constraint. If the resulting link stress is not satisfactory, we run the MDLB algorithm and relax the link stress constraints. Next, if the MDLB algorithm still fails, we run the BDML again with a relaxed diameter constraint. In this way, we can find a spanning tree with good characteristics in both diameter and link stress. We can adjust the relaxation steps to bias towards the metric we are more interested. In Section 6, we compare the properties and performance of these variations.

5.2. Reducing Bandwidth Consumption

We adopt a history-based approach to reduce bandwidth consumption. If the quality value to be reported to its parent at a node for a segment is similar to the value reported in the previous round, then this information will not be included in the reporting packet and the parent will use the value reported by this child in the previous round. By “similar” we mean the two values are equal within a small error interval, or both values are greater than an application specific lower bound threshold B , which denotes the lowest acceptable quality value. By lowering B we can further reduce the bandwidth consumption.

To implement such a strategy, we maintain a segment-neighbor table at each node. In this table, each row corre-

sponds to a segment. The table has $2c + 1$ columns, where c is the number of neighbors of the node in the spanning tree, including its parent and children. For every segment the node records the quality value received from and sent to each neighbor in the previous round. The other column is used to store the quality value inferred locally. Initially the table contains all zeros. Figure 6 shows the tables and their relationships.

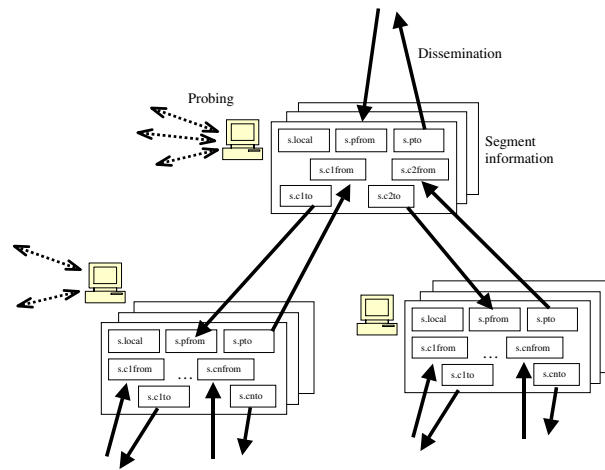


Figure 6. The segment-neighbor tables.

We use $s.local$, $s.cifrom$, $s.cito$, $s.pfrom$, and $s.pto$ to denote respectively the quality values of segment s inferred at the local node, received from the i th child, sent to the i th child, received from the parent, and sent to the parent.

On sending a packet to the parent, a node calculates the maximum quality value of all $s.cifrom$ and $s.local$ for every segment s , adds this value to the packet when the value is not similar to $s.pto$, and updates $s.pto$. On sending a packet to the x th child, a node calculates the maximum quality value of all $s.cifrom$, $s.local$ and $s.pfrom$ for every segment s , adds this value to the packet when the value is not similar to $s.ccto$, and updates $s.ccto$.

To ensure the segment information will not be disseminated if there is no node in the upper/lower subtree probing the segment or the quality value does not change, we add the following operations. On sending a packet to the parent, a node sets $s.pfrom$ equal to $s.pto$, so that even if the parent does not send down new values of this segment because there are no value changes, the node can still get the same value from $s.pfrom$ on receiving a packet from the parent. Similarly we set the value of $s.ccto$ on receiving a packet from child x , and set the value of $s.cxfrom$ on sending a packet to child x , and set the value of $s.pto$ on receiving packet from the parent. The pseudo code of the algorithm is shown in [17].

We show that all nodes achieve the best quality approximation for all segments when the above algorithm terminates. Obviously a node sends quality value upward only

when there are some changes of the segment quality in its subtree, in that case the value sent is the highest value currently in the subtree. So the root will get the highest lower bound for each segment, either inferred locally, or from a packet sent from one of its children, or from a stored value which denotes the value is sent from a child and there is no quality change in the subtree of that child. In either case the value is up-to-date. The root or a parent node will send the highest lower bound to its children, or the stored value indicates a child has inferred the same value. In either case it ensures the children will have the up-to-date highest lower bound.

6. Performance Evaluation

As a case study, we developed a path loss-state monitoring system based on our inference algorithms and implementation strategies. We use a packet-level simulator to study its performance in terms of inference accuracy, worst case link stress, and per-link bandwidth consumption.

6.1. Simulation Setup

We study the overlay network problems on real Internet topologies. We use three different topologies. Two of them are ISP-level topologies provided by the Rocketfuel project [16]. They reflect the typical ISP topologies around the year of 2002. The other one is an AS-level topology provided by NLANR [12]. It reflects the Internet topology at the AS-level around May of 2000.

We randomly select vertices in the topologies as overlay nodes. The size of the overlay networks varies from 4 to 256, with an exponential step in power of 2. For each size we generate 10 overlay networks with different random seeds. The performance evaluation results reflect the average values in the 10 overlay networks, unless specified otherwise. We use Dijkstra's shortest path algorithm to construct physical paths between pairs of overlay nodes. Only one ("rfb315" with 315 vertices) of the ISP topologies provides the link weight information. In the other ISP topology ("rf9418" with 9418 vertices) and the AS topology ("as6474" with 6474 vertices) we use physical hops as link weight in path construction.

In each configuration we conduct 1000 probing rounds and report the spatial statistics within one round or the temporal statistics for all rounds. The size of the quality information of one segment is set to 4 bytes. This size can be reduced to two bytes plus one bit if using loss bitmap.

6.2. Inference Accuracy

The system we implemented is a path loss-state monitoring tool based on our minimax inference algorithm. We use

the LM1 model in [13] to set the loss rate. The loss rate of the good nodes is set between 0 and 1%, and the loss rate of the bad nodes is set between 5% to 10%. The f parameter, or the fraction of the good nodes, is set to 90%. Nodes in the overlay network are randomly assigned to be in either good or bad states.

Our algorithm is a conservative one, of which the goal is to exclude bad paths from consideration, which is a fairly common operation in overlay nodes under normal conditions. Our algorithm guarantees it will detect every path that is in loss state. In other words, our system has a perfect error coverage, which was verified by the simulation results. As its cost, the system has a relatively high false positive rate (the ratio of the numbers of detected lossy path over the real lossy path in each round), as shown in Figure 7. The figure compares the Cumulative Distribution Function of false positive rate on 4 test configurations, namely, a 64-node overlay network on each of the three topologies and another 256-node overlay network on the topology "as6474". The number of probing packets is set to the size of a minimum segment set cover. The probing fraction is the ratio of the number of probed paths over the number of total $n \times (n - 1)$ paths. The figure shows under this condition all of them have high false positive rate in most probing rounds, which means each node will obtain a number of lossy paths that is several times higher than the real number. For example, in "as_64" and "rf9418_64", if the real number of lossy segments is 1, then in more than 60% of the probing rounds, a node will receive information denoting there are more than 4 lossy paths in the system.

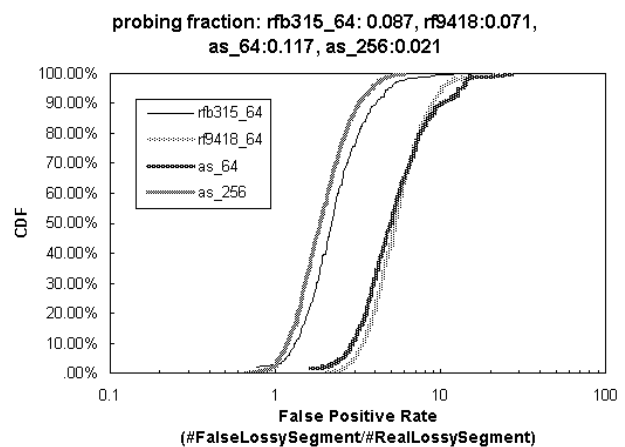


Figure 7. The CDF of false positive rate in 1000 probing rounds.

At first glance one may consider such inferences so inexact as to be of little use. However, we point out that our goal is not to identify all the lossy paths, but rather find as many as possible loss-free paths with high confidence and

low overhead. In this sense, our algorithm is useful since by probing a small set of paths we can still find many good paths with guaranteed quality, at least in the cases where the loss rate is not too high. Figure 8 shows, except in “rf9418_64”, the algorithm will identify more than 80% of the good paths with less than 10% paths probed in most probing rounds. Even in “rf9418_64”, the good path detection rate is still higher than 60% in most rounds.

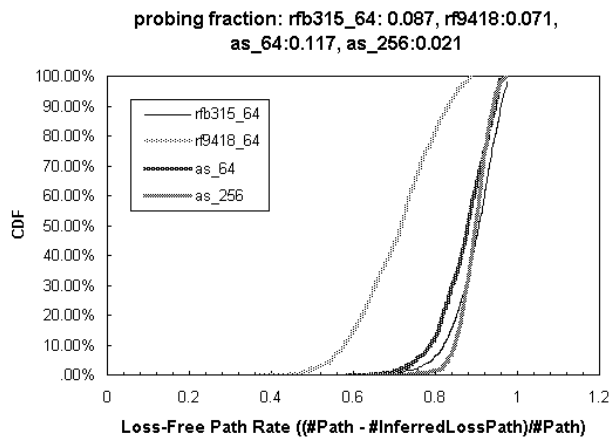


Figure 8. The CDF of good path detection rate in 1000 probing rounds.

6.3. Link Stress and Bandwidth Consumption

We study the worst-case link stress and bandwidth consumption in this subsection. We compare the performance of several tree-building algorithms on the metrics of average and worst-case link stress, tree diameter, and the corresponding link bandwidth consumption in our system. The algorithms corresponds to the diameter constrained minimum spanning tree (DCMST), the minimum diameter link stress bounded spanning tree (MDLB), the limited diameter, link stress balanced spanning tree (LDLB), and the combination of the MDLB and a bounded diameter, minimum link stress spanning tree, with various increasing steps (MDLB+BDML1, MDLB+BDML2). The resulting link stress statistics and the bandwidth consumption values on the topology “as_64” are shown in Figure 9.

Figure 9 shows that all of these trees have small average link stress. However, their worst-case link stress, which may affect the system robustness and performance bottleneck, is different. Not surprisingly, the algorithm oblivious to link stress (DCMST) has the worst performance in terms of stress balance and link bandwidth consumption. It generates a worst-case link stress of 61 and a corresponding bandwidth consumption of about 300KB. In MDLB algorithm, we set the initial link stress limit $r_{max}(e)$ be 1 for

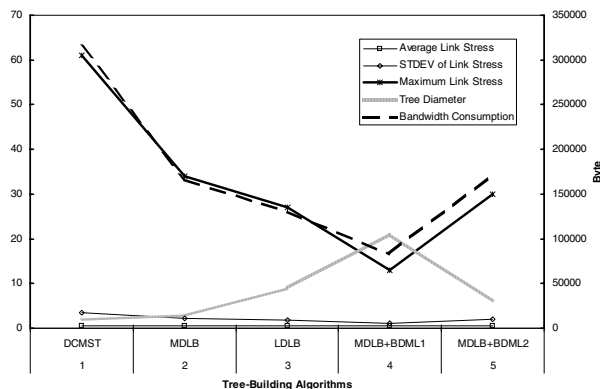


Figure 9. The relationship between link stress, diameter, and bandwidth consumption.

every link e , and run the algorithm to minimize the tree diameter. If no such tree with these link stress constraints can be constructed, we increment $r_{max}(e)$ by 1 for every link e and repeat the algorithm until one tree is found. Since the MDLB algorithm solves the problem by periodically relaxing constraint on link stress, the final link stress may not be reduced much. The results show the worst-case link stress is 33 in this case. In LDLB algorithm, we set the diameter limit be $2 \log n$ where n is the number of overlay nodes, and run the algorithm to balance the link stress. We do this by selecting the path at each step that satisfies the diameter constraint while minimizing the maximum link stress. In this case, the worst-case link stress is lower at 27. The combined algorithm can achieve either low link stress or diameter by adjusting the steps it uses for recalculating constraints. For example, the MDLB+BDML1 algorithm increases the link stress constraints by 1 at each step while relaxing the diameter constraint by $\log n$. It can reduce the worst-case link stress to 13, but incur a large diameter. On the other hand, the MDLB+BDML2 algorithm increases the link stress constraints by 1 while relaxing the diameter constraint by 0.1. It has comparable performance as of LDLB. From the figure we also see the worst-case bandwidth consumption is highly correlated to the worst-case link stress in our system.

6.4. Reduction of Bandwidth Consumption

Finally, in Figure 10 we show the bandwidth consumption for quality information dissemination on the topology “as_64”. In each round the bandwidth required for any on-tree link is usually small, typically a few kilobytes or less. As shown in Figure 10, our history-based algorithm can substantially reduce the information required to be transmitted. The average bandwidth consumption on each link is reduced from 3KB to around 2.6 KB. The reduction is determined by link loss-state changes in successive rounds.

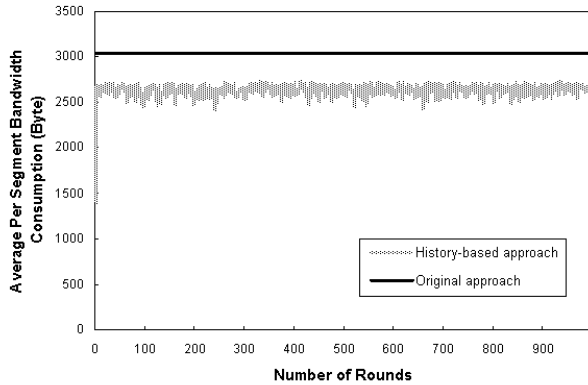


Figure 10. The bandwidth consumption.

7. Conclusion

In this paper we described a distributed implementation of our overlay path probing and inference algorithms. Our algorithms exploit the topology information to infer the path quality from the probing results of other paths. The approach is based on the fact that in a sparse network overlay links significantly overlap with one another so that a path probing reveals not only the quality of the specified path but also partial quality information of other overlapping paths. In our system, each overlay node selects a subset of paths to probe. The combined probing results can be used to infer the quality information of all paths while the total number of probings is much lower than the number of paths.

In our implementation, we addressed the problem of performance bottleneck, link stress balancing, bandwidth consumption reduction, and local quality information availability. We employed a minimum diameter, link stress bounded overlay spanning tree to disseminate quality information. Every overlay node participates in the path probing and information dissemination process. The quality inference is incrementally refined at each node. At the end of each probing round all the nodes obtain the best approximation of the path quality information. To reduce the bandwidth consumption we utilized history data of the quality values. Simulation results show our implementation can achieve good performance in probing overhead reduction and load balancing while maintaining good quality inference accuracy.

Acknowledgements

This work was supported in part by the U.S. Department of the Navy, Office of Naval Research under Grant No. N00014-01-1-0744. This work was also supported in part by National Science Foundation grants CCR-9912407, EIA-0000433 and EIA-0130724.

References

- [1] A. Abdalla, N. Deo, and P. Gupta. Random-tree diameter and the diameter constrained MST. *Congressus Numerantium*, 144:161–182, 2000.
- [2] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient Overlay Networks. In *Proceedings of 18th ACM SOSP*, 2001.
- [3] T. Bu, N. Duffield, F. L. Presti, and D. Towsley. Network tomography on general topologies. In *Proceedings of ACM SIGMETRICS*, 2002.
- [4] V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.
- [5] M. Coates, R. Castro, and R. Nowak. Maximum likelihood network topology identification from edge-based unicast measurements. In *Proceedings of ACM SIGMETRICS*, June 2002.
- [6] M. Coates, A. Hero, R. Nowak, and B. Yu. Internet tomography. *IEEE Signal Processing Magazine*, 19(3):47–65, May 2002.
- [7] W. Cui, I. Stoica, and R. H. Katz. Backup path allocation based on a correlated link failure probability model in overlay networks. In *Proceedings of 10th IEEE International Conference on Network Protocols (ICNP'02)*, November 2002.
- [8] N. Duffield, F. Presti, V. Paxson, and D. Towsley. Inferring link loss using striped unicast probes. In *Proceedings of IEEE INFOCOM*, April 2001.
- [9] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the Internet topology. In *Proceedings of ACM SIGCOMM*, pages 251–62, September 1999.
- [10] Y. Huang and P. K. McKinley. Switch-aided flooding operations in ATM networks. In *Proceedings of IEEE INFOCOM*, April 1997.
- [11] M. Kwon and S. Fahmy. Topology-aware overlay networks for group communication. In *Proceedings of The 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2002)*, 2002.
- [12] NLANR. National laboratory for applied network research. <http://moat.nlanr.net/ Routing/rawdata>, 2000.
- [13] V. N. Padmanabhan, L. Qiu, and H. J. Wang. Server-based inference of Internet link lossiness. In *Proceedings of IEEE INFOCOM*, April 2003.
- [14] A. Shaikh, M. Goyal, A. Greenberg, R. Rajan, and K. Ramakrishnan. An OSPF topology server: Design and evaluation. *Journal of Selected Areas in Communications: Special Issue on Recent Advances on Fundamentals of Network Management*, May 2002.
- [15] S. Shi and J. S. Turner. Routing in overlay multicast networks. In *Proceedings of IEEE INFOCOM*, June 2002.
- [16] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP topologies with Rocketfuel. In *Proceedings of ACM SIGCOMM*, August 2002.
- [17] C. Tang and P. K. McKinley. A distributed approach to topology-aware overlay path monitoring. Technical Report MSU-CSE-03-24, Computer Science and Engineering, Michigan State University, East Lansing, Michigan, August 2003.
- [18] C. Tang and P. K. McKinley. On the cost-quality tradeoff in topology-aware overlay path probing. In *Proceedings of International Conference on Network Protocols (ICNP)*, pages 268–279, November 2003.
- [19] Y. Tsang, M. Coates, and R. Nowak. Passive network tomography using EM algorithms. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, May 2001.
- [20] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker. On the constancy of Internet path properties. In *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, November 2001.