This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the ICC 2008 proceedings.

On the Investigation of Path Preference in End-to-End Network Measurements

Xing Jin Qiuyan Xia S.-H. Gary Chan
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
Email: {csvenus, xiaqy, gchan}@cse.ust.hk

***Abstract*— Overlay applications have used various tools to measure path properties in order to construct efficient overlay networks. Typical examples include delay measurement, connectivity measurement and residual bandwidth measurement. While different tools have different measurement targets, it is difficult to design general criterion for selecting paths to measure. In this paper, we study path properties and figure out some simple rules for path selection in measurements. We first examine the relationship between path delay and residual bandwidth. Our measurement results indicate that short paths often have high residual bandwidth. We then impose high preference on short paths in traceroute measurements. The results show that traceroute measurements with path preference achieve comparable inference efficiency and much lower overhead as compared to traditional traceroute measurements. Our study suggests that we can preferentially select short paths to measure. This can fulfill various requirements on path properties.**

I. Introduction

With the quick development of network technologies, overlay networks have been widely used in various applications. Examples include application-layer multicast (ALM), peer-to-peer (P2P) audio and video streaming, and overlay path routing. According to CacheLogic Research, P2P traffic, which first emerged in 1999 with Napster, has accounted for as much as 70% Internet traffic in 2006.

In order to build an efficient overlay network, the knowledge of the underlay topology is important. For example, two seemingly disjoint overlay paths may share common underlay links; therefore the selection of overlay paths without the knowledge of underlay may lead to serious link congestion. Currently, there are mainly three types of tools for end-to-end network inference: (1) Delay measurement using ping: For example, Narada uses the ping tool to select a close parent for a new joining host [1]. (2) Connectivity measurement using traceroute: For example, TAG uses traceroute to infer the path connectivity among hosts, and constructs a tree with low delay and low stress [2]. (3) Residual bandwidth measurement using tools like Pathload [3]: For example, Overcast measures residual path bandwidth to build a high-bandwidth tree [4].

Clearly, different tools have different measurement targets. When ping is used, we often want to identify a short path with low delay. When Pathload is used, we want to identify a path with high residual bandwidth. When traceroute is used, our target is to discover as many underlay links as possible. Given different measurement targets, it is difficult to design general criterion for selecting paths to measure.

In this paper, we study path properties and figure out some simple rules for path selection in topology inference. Our motivation comes from the observation that paths within the same local networks often have both low delay and high residual bandwidth, while paths crossing multiple ISPs often have high delay and relatively low residual bandwidth. We hence study the relationship between path delay and residual bandwidth. We conduct measurements on the PlanetLab testbed [5]. Our measurement results confirm that short paths often have higher residual bandwidth than long paths. This suggests to preferentially select short paths to measure.

We then study the impact of the path preference to traceroute measurements. We impose high preference on short paths and integrate path preference into Max-Delta, a traceroute-based topology inference scheme [6]. We compare the performance of Max-Delta with and with no path preference on different network topologies. Our results show that the two schemes achieve comparable inference efficiency. In addition, Max-Delta with path preference achieves much lower measurement overhead as it prefers short paths.

In summary, our study shows that it is beneficial to preferentially select short paths to measure in topology inference. This can fulfill various requirements on path properties. Hence, in network measurement, we can first estimate path delay, and then preferentially select short paths to conduct traceroute or residual bandwidth measurement. Note that ping measurement incurs much lower traffic than traceroute and residual bandwidth measurement. We can hence estimate path delay with low overhead. We may also use coordinate estimation tools like Global Network Positioning (GNP) to estimate path delay.

The rest of the paper is organized as follows. In Section II, we study the relationship between path delay and residual bandwidth. In Section III, we compare the performance of Max-Delta with and with no path preference. Finally, we conclude in Section IV.

II. Revisiting Path Properties

A. Delay and Residual Bandwidth

An end-to-end path may be characterized by many metrics, e.g., delay, hop count, loss rate and residual bandwidth. Among them, end-to-end path delay is the most commonly

This work was supported, in part, by Direct Allocation Grant at the HKUST (DAG05/06.EG10), and Hong Kong Research Grant Council (611107).

978-1-4244-2075-9/08/$25.00 ©2008 IEEE

used in overlay applications. Path delay is often computed in terms of round-trip time (RTT), which can be obtained through the ping program. Given a pair of source and destination hosts, the ping program sends an Internet Control Message Protocol (ICMP) request message from the source to the destination, expecting an ICMP reply message to be returned. The program usually sends multiple request messages. Among all the returned results, the minimum RTT is used to compute path delay.

Ping measurement has been widely used in overlay construction, e.g., NICE, Narada, DT and ALMI. For example, in Narada, a host keeps selecting some random hosts to ping [1]. It adds a new path into the overlay network if the new path is short enough. It also drops long paths from the overlay network. After the overlay is constructed, a shortest path tree is built on top of the overlay for data delivery. Similarly, many other applications endeavor to identify short paths with small RTT between hosts.

Another popular path metric is residual path bandwidth. The residual bandwidth of a path is defined as the difference between the path capacity and the amount of bandwidth already taken by all active and backup traffic traversing the path. Many tools have been proposed to measure residual path bandwidth, e.g., Pathload, PTR/IGI, TOPP, Pathchirp and Spruce.

If we know the residual bandwidth of different paths, we can select the ones with high residual bandwidth for overlay construction. As a result, the data transmission rate between hosts is high. A typical example is Overcast, which aims to build an overlay tree with high transmission rate [4]. In Overcast, when a new host arrives, it first estimates its bandwidth to the root and to each of the root's children. If the bandwidth to any of these children is close to the bandwidth to the root, the new host moves one level down to this child and repeats the process. This procedure continues until the current host has no children with satisfactory bandwidth to the new host. Then the current host becomes the new one's parent.

As discussed, in order to achieve low end-to-end delay, short paths with small RTT are preferred. On the other hand, in order to achieve high transmission rate, paths with high residual bandwidth are preferred. Hence, we have two orthogonal metrics for path selection. Fortunately, we find that these two metrics are not fully independent or contradictive. According to [7], the receiver of a friendly TCP connection can use the following function to calculate its expected bandwidth:

$$B = \frac{s}{RTT\sqrt{\frac{2p}{3}} + RTO\left(3\sqrt{\frac{3p}{8}}\right)p(1+32p^2)}.$$

This gives the TCP throughput B in bytes/s, as a function of the packet size s, round-trip time RTT, steady-state loss rate p, and the TCP retransmit timeout value RTO. The function indicates that B is a reciprocal function of RTT. In other words, a path with small RTT often has high residual bandwidth, and vice versa.

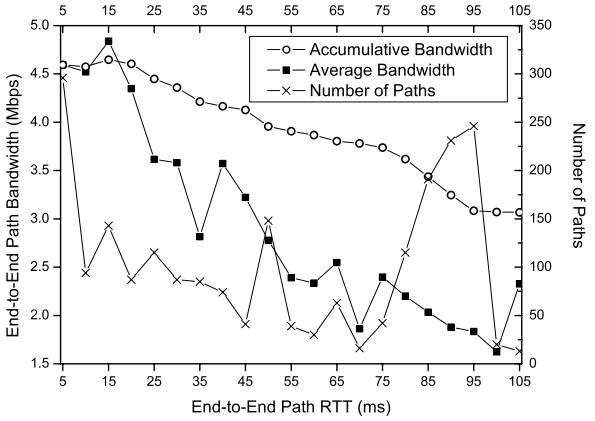


Fig. 1. Residual bandwidth versus RTT of paths (on PlanetLab).

In order to verify the relationship between path RTT and residual bandwidth, we have done the following measurement study on the PlanetLab testbed.

B. PlanetLab Measurement Study

We have obtained a PlanetLab topology through end-to-end measurements. We randomly select a number of hosts from PlanetLab and conduct all-pairs traceroutes between them. Due to network and host dynamics (some hosts may unexpectedly fail during our measurements), a small portion of the traceroutes cannot be completed. From the traceroutes obtained, we construct a topology consisting of 72 hosts and 2540 overlay paths (we assume that paths are symmetric and only measure one path between a pair of hosts).

We also measure residual bandwidth of paths. In the measurement, a sender sends a data clip to a receiver, and the receiver measures the downloading time. The path bandwidth is computed as the data clip size divided by the downloading time. A data clip contains $1,000$ packets and one packet is set to $1,000$ bytes. Each host measures its bandwidth to all the other hosts. Among the 72 hosts, we obtain the bandwidth results on 2189 paths (some transfer cannot be completed due to network and host dynamics). The maximum and minimum bandwidth is about 14.4Mbps and 35Kbps, respectively, and over 94% paths have bandwidth higher than 1Mbps. We combine the traceroute results and bandwidth measurement results to build a testbed topology, with 72 hosts, 2176 overlay paths, 1145 underlay links and 627 routers.

Figure 1 shows the relationship between residual bandwidth and RTT of paths. We divide path RTT into a few intervals. The intervals start from 0ms and each interval spans 5ms. For example, in the curve denoted "Number of Paths", a point $(5, 296)$ means that there are 296 paths whose RTT falls into the interval $[0\text{ms}, 5\text{ms})$. Similarly, a point $(10, 94)$ means that there are 94 paths whose RTT falls into the interval $[5\text{ms}, 10\text{ms})$. However, the last point in the curve has different

meaning. The point $(105, 13)$ in the curve means that there are in total 13 paths whose RTT is no less than 100ms. Note that other points in the curve have shown the distribution of paths whose RTT is less than 100ms.

The curve denoted "Average Bandwidth" in Fig. 1 shows the average path bandwidth in different RTT intervals. With the increase of RTT, average path bandwidth generally decreases. As shown, there are a few paths with significantly high residual bandwidth (higher than 4.5Mbps) and low RTT (less than 15ms). These are mainly paths within the same campus network. In the current PlanetLab, a university usually contributes $2 - 4$ computers as PlanetLab hosts. These computers are often close to each other and have high bandwidth connections between each other. As the last point in the curve shows, the average bandwidth of paths whose RTT is higher than 100ms is 2.3Mbps. This is unexpectedly higher than its previous points. However, as there are only 13 paths whose RTT is higher than 100ms, the path number is too small and the result lacks enough representativeness.

The curve denoted "Accumulative Bandwidth" shows the average bandwidth of paths in an accumulative RTT range. For example, the point $(10, 4.58)$ in the curve means that the average bandwidth of paths whose RTT is less than 10ms is 4.58Mbps. Unlike other points, the last point $(105, 3.06)$ means that the average bandwidth of all paths is 3.06Mbps. As shown, when we take long paths into computation, the average residual bandwidth decreases.

The above measurement results show that a short path often has relatively high residual bandwidth, while a long path often has relatively low residual bandwidth. This suggests us to preferentially use short paths for overlay construction, since they can provide low end-to-end delay and high transmission rate.

III. Considering Path Preference in Traceroutes

A. Comparison Schemes

We now study the impact of path preference to traceroute measurements. We select Max-Delta as the baseline scheme for verification [6]. Max-Delta is proposed to efficiently infer the topology among a group of hosts by traceroute. It works as follows: Hosts first utilize light-weight tools such as GNP to estimate their network coordinates, and report them to a central server. The following inference procedure is divided into multiple iterations. In each iteration, the server selects a target for each host to traceroute. Hosts then traceroute their targets and report the results to the server. The server combines all the results obtained in the iteration and based on that, starts the next iteration on target assignment. Such process is repeated until a certain stop rule is achieved (e.g., reaching a certain number of iterations).

A target is selected as follows. For a certain host A, suppose that the path between A and another host B has not been measured. The server computes the distance between A and B in the discovered topology $D_p(A, B)$, using shortest path routing. The server also computes the distance between them based on their coordinates, $Euclidean(A, B)$. If coordinate estimation is accurate, $Euclidean(A, B)$ will approximate the real network distance between A and B. Define the gap between these two values as

$$\Delta(A, B) = D_p(A, B) - Euclidean(A, B).$$

If $\Delta(A, B)$ is large, it is with high probability that some links between A and B (leading to a shorter path in the discovered topology) are not discovered yet. For all unmeasured paths between A and other hosts, the server selects the path with the maximum Δ value as A's traceroute target.

TABLE I

Unmeasured paths adjacent to a host.

Path	$Euclidean$ (ms)	Δ (ms)
1	55	25
2	95	30
3	85	50
4	130	70

To integrate path preference into Max-Delta, we modify the target selection mechanism as follows. We first define a few intervals for path delay. The intervals start from 0ms and each interval spans $t$ms, where t is a system parameter. An unmeasured path is put into a certain interval according to its $Euclidean$ value. We then examine the intervals according to their starting values in an ascending order. For a certain interval, we examine the unmeasured paths falling into it, and select the one with the maximum Δ value. If there exists such a path, the path is selected as the traceroute target in the next iteration. Otherwise, the interval does not contain any unmeasured paths. We then move to the next interval with larger starting value, and repeat the above process. In the following, we call this scheme *distance limited Max-Delta*, or simply *DLMD*. Its work flow is shown in Algorithm 1.

We show an example to illustrate Max-Delta and DLMD. Table I shows the set of unmeasured paths adjacent to a host in a certain iteration. In Max-Delta, the host selects the path with the maximum Δ value as the traceroute target, i.e., path 4. If we use DLMD, we put the paths into different intervals according to their $Euclidean$ values. Suppose $t = 50$ms. We sequentially examine the intervals. The first interval $[0\text{ms}, 50\text{ms})$ does not contain any paths. We then move to the second interval $[50\text{ms}, 100\text{ms})$. There are three paths in the interval, i.e., paths 1, 2 and 3. From these three paths, we select the one with the maximum Δ value, i.e., path 3. Hence, path 3 is selected as the traceroute target in the next iteration.

B. Results on Internet-like Topologies

We evaluate Max-Delta and DLMD through simulations. We generate a number of *Transit-Stub* topologies with GT-ITM [8]. Each topology is a two-layer hierarchy of transit networks and stub networks. A topology contains $6,000$ routers and around $42,000$ links. We randomly put 500 hosts into the network. Each host is connected to a unique stub router with 1ms delay, while the delay of core links is given by the topology generator. In DLMD, we set $t = 50$ms. We use the following metrics to evaluate an inference scheme.

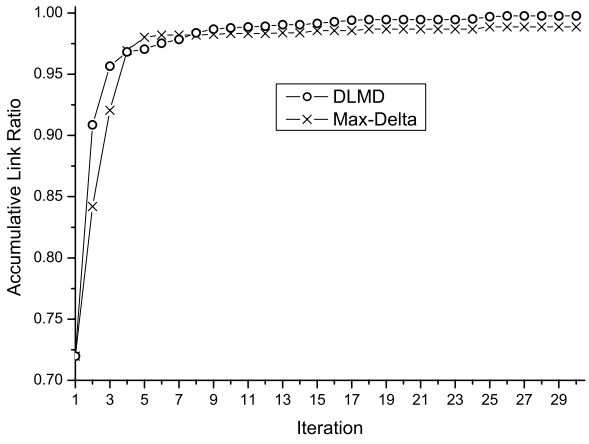
Algorithm 1 : TARGET SELECTION FOR HOST i IN DLMD
INPUT: t - interval span
E_{max} - the maximum path delay in theory (a constant, assuming E_{max} is larger than all $Euclidean$ values)
S_i - set of unmeasured paths adjacent to i
OUTPUT: A path from S_i as the next traceroute target

```
1: p ← ⌈E_max/t⌉
2: for j ← 0 to p − 1 do
3:     set I_j ← ∅
4: end for
5: for all path t − i in S_i do
6:     k ← ⌊Euclidean(t, i)/t⌋
7:     put path t − i into set I_k
8: end for
9: for j ← 0 to p − 1 do
10:    if I_j ≠ ∅ then
11:        select path m − i from set I_j s.t.
           Δ(m, i) = max{Δ(x, i)|∀ path x − i ∈ I_j}
12:        output path m − i
13:        return
14:    end if
15: end for
```

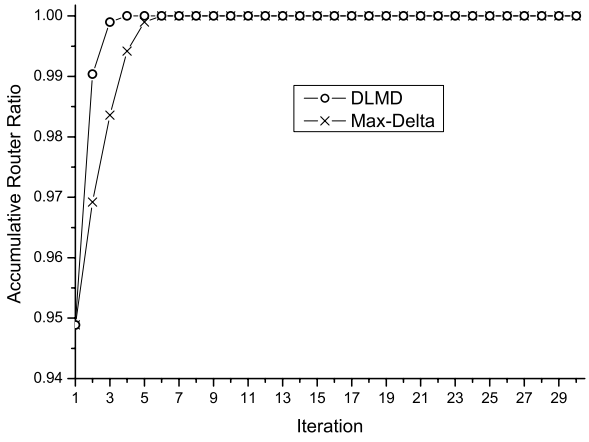
- *Link ratio*: defined as the ratio of the number of links in the inferred topology to the total number of links in the actual underlay topology [6].
- *Router ratio*: defined as the ratio of the number of routers in the inferred topology to the total number of routers in the actual underlay topology [6].
- *Resource usage*: Given a traffic between two points, the resource usage is computed as the size of traffic packets times the delay between the two points. We compute the resource usage of an inference scheme as the total resource usage for traceroute measurements in the inference. The resource usage of a single traceroute can be computed as in [9].

Figure 2 compares the performance of the two inference schemes. Figure 2(a) shows the accumulative link ratios achieved by the schemes. Max-Delta can quickly discover over 95% links, after 4 iterations. Its link ratio then converges around 99%. DLMD achieves comparable performance as Max-Delta. It discovers over 95% links only after 3 iterations. Its link ratio is higher than Max-Delta's in the first several iterations and in the iterations after the 8th. Figure 2(b) shows the accumulative router ratios achieved by the schemes. DLMD performs slightly better than Max-Delta. Max-Delta can discover over 99% routers after 4 iterations, and DLMD achieves it only after 2 iterations. From these two figures, we see than DLMD achieves slightly higher inference efficiency than Max-Delta.

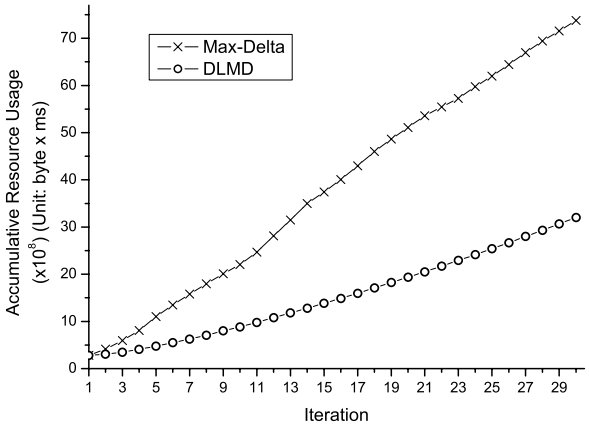
Figure 2(c) shows the accumulative resource usage achieved by the schemes. As expected, DLMD achieves much lower resource usage than Max-Delta. DLMD preferentially selects short paths to traceroute. Hence, its traceroute averagely sends less probing packets than that in Max-Delta, and these packets traverse shorter distances. As shown, after 30 iterations, the resource usage of DLMD is only 43% of that of Max-Delta. Hence, DLMD can significantly reduce the measurement over-



(a)



(b)



(c)

Fig. 2. Performance comparison of different inference schemes. (a) Accumulative link ratio. (b) Accumulative router ratio. (c) Accumulative resource usage.

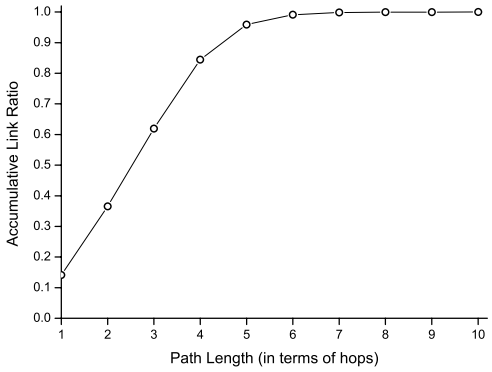


Fig. 3. Accumulative link ratio by paths no more than certain length.

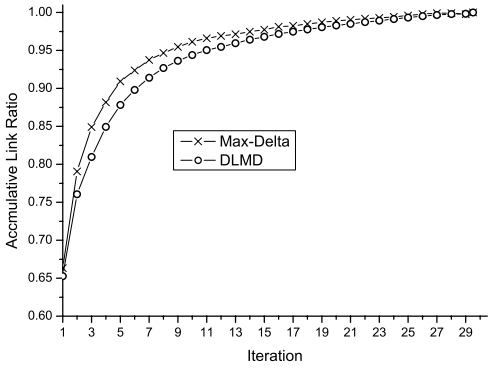


Fig. 4. Accumulative link ratios achieved by the schemes (on PlanetLab topology).

head.

The above results show that DLMD achieves higher inference efficiency than Max-Delta. This contradicts the intuition that short paths only contain a few links and routers, and hence only a few undiscovered links or routers. We examine our GT-ITM topologies to explain DLMD's good performance. Given the number of hosts $N = 500$, all-pairs traceroutes between hosts result in $N(N-1)/2 = 124,750$ paths (assuming paths are symmetric). We analyze the properties of the paths in a typical simulation. In the simulation, there are in total $1,677$ different links and $1,055$ different routers in the paths.

Figure 3 shows the accumulative link ratio achieved by paths no more than certain length. For example, path length 5 corresponds to 95.9% in the curve. It means that if we only measure paths whose length is less than or equal to 5, we can discover 95.9% links on the underlay. When path length is 10, such value is 100%. It means that we can cover the whole underlay topology using paths whose length is no more than 10. The figure shows that short paths have contained most of underlay links. We can discover almost all links by only tracerouting short paths. Therefore, when we integrate path preference into Max-Delta and prefer short paths in inference, we do not sacrifice inference efficiency.

C. Result on PlanetLab Topology

We further evaluate the inference schemes on the PlanetLab topology. We select 60 hosts from the topology and neglect traceroutes among them that contain anonymous routers. By careful selection of the hosts, the remaining topology consists of around 96% of the pairwise paths. We use this topology as the complete underlay topology among the hosts. Figure 4 shows the link ratios achieved by the schemes on the PlanetLab topology. Both Max-Delta and DLMD can quickly discover underlay links. Max-Delta performs slightly better than DLMD. To achieve a certain link ratio, Max-Delta needs less iterations than DLMD. For example, to discover 95% links, Max-Delta needs 9 iterations, while DLMD needs 11 iterations. As the performance gap between the two schemes is not large, we conclude that DLMD's performance is comparable with Max-Delta's.

IV. Discussion and Conclusion

There are various ways to measure path properties in topology inference. In this paper, we try to figure out some simple rules for path selection in order to fulfill various measurement targets. Our study shows that it is beneficial to preferentially select short paths to measure. Short paths often have high residual bandwidth. And traceroute measurements with such path preference can reduce measurement overhead, at negligible penalty in inference efficiency. On the other hand, the real Internet is complicated. A short path does not necessarily have higher residual bandwidth than a long path. And in traceroute measurements, it is not always efficient to select short paths to measure (as shown in Fig. 4). We hence cannot select paths to measure only based on their delay. We need to carefully design how to assign preference to short paths in topology inference.

References

[1] Y. H. Chu, S. Rao, S. Seshan, and H. Zhang, "A case for end system multicast," *IEEE JSAC*, vol. 20, no. 8, pp. 1456–1471, Oct. 2002.

[2] M. Kwon and S. Fahmy, "Topology-aware overlay networks for group communication," in *Proc. ACM NOSSDAV'02*, May 2002, pp. 127–136.

[3] M. Jain and C. Dovrolis, "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput," in *Proc. ACM SIGCOMM'02*, Aug. 2002, pp. 295–308.

[4] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole, "Overcast: Reliable multicasting with an overlay network," in *Proc. OSDI'00*, Oct. 2000, pp. 197–212.

[5] PlanetLab. [Online]. Available: http://www.planet-lab.org

[6] X. Jin, W.-P. K. Yiu, S.-H. G. Chan, and Y. Wang, "Network topology inference based on end-to-end measurements," *IEEE JSAC*, vol. 24, no. 12, pp. 2182–2195, Dec. 2006.

[7] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: a simple model and its empirical validation," in *Proc. ACM SIGCOMM'98*, Sept. 1998, pp. 303–314.

[8] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proc. IEEE INFOCOM'96*, March 1996, pp. 594–602.

[9] X. Jin, Q. Xia, and S.-H. G. Chan, "A cost-based evaluation of end-to-end network measurements in overlay multicast," in *Proc. IEEE INFOCOM MiniSymposium'07*, June 2007, pp. 2581–2585.