

# QoEScope: Adaptive IP Service Management for Heterogeneous Enterprise Networks

Yueping Zhang, Vishal Singh, Yu Gu, and Guofei Jiang  
NEC Laboratories America, Inc.  
4 Independence Way, Princeton, NJ 08540  
Email: {yueping, vishal, yugu, gfj}@nec-labs.com

Yu Ru  
University of Illinois at Urbana Champaign  
1308 West Main Street, Urbana, IL 61801  
Email: yuru2@uiuc.edu

**Abstract**—In the recent years, a progressively growing number of computing and communication services have undertaken the migration from their conventional media to the new unified platform, IP networks. As a consequence, business success of service providers becomes largely determined by the effectiveness of their service management schemes, which require rapid identification of problems and resolution of network-related anomalies. However, this is a non-trivial task in heterogeneous enterprise networks due to service providers’ invisibility of the health and performance of the underlying carrier network. In addition, the gap between quality of service (QoS) measurements reflecting network performance and quality of experience (QoE) metrics indicating user-perceived service quality further makes effective service management more challenging. In this paper, we present a unified service management system called QoEScope, which combines *scalable* end-to-end probing, *accurate* topology inference in the presence of implicit routers, *adaptive* bridging between QoS measurement and QoE metrics, and *intelligent* root cause analysis. Extensive testbed emulations and Internet experiments demonstrate that QoEScope is a highly practical and effective IP service management solution for heterogeneous enterprise networks.

## I. INTRODUCTION

There is an increasing trend of unified communication over Enterprise IP networks across the globe, and Voice over IP (VoIP) and NetMeetings have been widely deployed. From the perspective of IP service management, it is highly desirable to identify problematic network segments promptly and accurately when users perceive unsatisfying service qualities. There are a number of challenges in pinning down the exact network segments that cause poor user-perceived performance. First, IP service users dynamically join service sessions (e.g. IP video conferences) from different network locations. As a consequence, the underlying network topologies change across multiple service sessions. This creates a non-trivial obstacle for network tomography based methods [3], which usually assume static network environments and highly correlated measurements. Second, Enterprise IP services may span over vastly separated geographical locations with rented third-party networks, oftentimes in the form of virtual private networks (VPNs). As a consequence, the service providers have limited information about the underlying network and cannot know exactly the quality of service perceived by the end users. Third, the network condition is always changing. In particular, problems presented in one session may not present themselves

in another session and network status discovered in a single service session may not reflect the true underlying problematic network segments. This is especially true given the transient behaviors in today’s Internet. Meanwhile, even if measurements of the network can be performed, the measurements are subject to resource limitations from the underlying networks and the client side. Some of the measurements can only be obtained at a very coarse granularity, such as packet loss rate; some of them could be distorted, such as end-to-end delay; and some of them can contain missing data, such as routing information obtained from `traceroute`. Fourth, even if all the above information is available, inferring the problematic network segments from accurate end-to-end measurements can itself be a difficult problem that is highly *under-determined* [3].

Targeted at overcoming challenges listed above, we propose a framework called QoEScope, which identifies problematic network segments using an end-to-end approach. QoEScope consists of four key technical components/steps. First, we perform scalable end-to-end measurements of quality of service (QoS) metrics (e.g., packet loss rate, delay, and jitter) over the path. These measurements are taken using lightweight client-side Java applets, which can either be integrated with any third-party applications (e.g., Web browser) or run independently. Then, we introduce a network routing topology inference algorithm called *NetScan*, which complements information obtained from `traceroute` and achieves very high accuracy even when intermediate routers do not respond to `traceroute` queries. Third, we develop a QoE learning engine, which utilizes a neural network classifier to adaptively map QoS measurements obtained from active probing to user-perceived quality of experience (QoE). Finally, combining topology information from *NetScan* and user perceived qualities from the QoE learning engine, we devise a simple yet effective algorithm that locates the most possible problematic network segments.

We implemented QoEScope in real systems and performed extensive experiments both in controlled environments (i.e., our in-house testbed) and in the wild Internet (i.e., Planetlab [17]). All of our experiments demonstrate that our framework exhibits easy deployability without modification to applications’ existing code base, high accuracy topology inference even in case where all intermediate routers are implicit, and

real-time adaptability to dynamic network environments. However, we emphasize that this paper does not attempt to claim optimality of any of the four components in QoEScope and acknowledge that each of them has its own limitations. Instead, the purpose of this paper is to share with the community our experience of design, implementation, and evaluation of a *practical* QoE management tool (which is a combination of several simple yet effective technical components) for a real IP service system.

The rest of the paper is structured as follows. In Sections II–V, we respectively describe QoEScope’s end-to-end probing, topology inference, QoE learning, and problem reasoning mechanisms. In Section VI, we evaluate QoEScope’s performance in both controlled testbeds and the Internet. In Section VII, we give a brief review of related work. Finally in Section VIII, we conclude this paper and point out further work.

## II. END-TO-END PROBING

We start with the end-to-end probing mechanism used to obtain real-time monitoring of internal traffic dynamics inside the underlying carrier network. Before the conference session starts, each user clicks a URL, which signals the server to push a Java applet to the client’s Web browser. Then, the applet starts sending and receiving probing packets at a constant rate between the client and server. We set the packet size of each probe to 200 bytes and inter-packet interval to 50 ms<sup>1</sup>. At both the client and server sides, sequence number and timestamp of each incoming probe are recorded. At the end of every  $T$  seconds (we set  $T = 10$ , which is configurable), both the client and server calculate one-way delay, packet loss rate, and delay jitter for the path connecting them. We implemented a data processing mechanism that synchronizes clocks of the server and clients in a manner similar to NTP. Subsequently, the client sends the server a *report* message containing the calculated results. Then, the server combines this report with its own measurements and estimates QoE perceived by the client, i.e., the *transmission rating factor* (or R-value). Due to the complexity of E-model, we do not present its relevant equations in the paper, but refer interested readers to [10] for details. We also note that while the E-model involves many metrics that cannot be directly measured via end-to-end probing, in QoEScope we express the R-value as a function of delay and loss and set other metrics to their default values specified by [10]. This way, the service provider is able to monitor network condition and user-perceived quality for each client in real time.

We note that for the enterprise IP conference system that QoEScope is designed for, the size of participants in a conference session is usually less than 100, in which case the above probing mechanism imposes a minimum impact on the existing communication traffic. For larger-scale IP service systems, the probing overhead may become a major issue. For instance, using the packet size and inter-packet interval

<sup>1</sup>Those values are chosen according to the transport protocol employed in our test IP conference system. In practice, they can be changed to different values to emulate other protocols, e.g., RTP [20] and DCCP [13].

specified above, 1000 clients will generate *combined* probing traffic of 32 Mb/s, which, when aggregate at the server, may affect the existing IP services. This problem can be mitigated by incorporating various probing optimization techniques [11] or certain passive measurement schemes [18] in the existing literature. We do not seek to address this problem in the paper, but leave further exploration in this direction for future work.

## III. QOE LEARNING ENGINE

As described in the last section, we calculate the R-value purely based on QoS measurements (e.g., delay and loss) and set all other variables in the E-model to their default values, which however could deviate from their real values in practical systems. Moreover, user perceived quality of experience is a subjective and sensitive metric, which can be significantly affected by many external factors, such as level of room noise, loudness of the receiving speaker, mood of the listener, and even workload of the client’s machine. Therefore, it is a critical challenge for any practical IP service management system to correctly and dynamically infer users’ QoE.

In this paper, we tackle this challenge by taking a machine learning based approach. Specifically, we model the relationship between QoS and QoE metrics using a multilayer perceptron classifier (MLP). Due to limited space, we refer interested readers to [22] for details about the MLP classifier used in QoEScope. In QoEScope, training data is simply a collection of network measurements of delay, loss, and jitter. We first train the classifier using R value computed based on QoS measurement data and then *refine* it with labeled data derived from real users’ MOS feedbacks. In our prototype, for instance, each client can use a web page to report QoE feedbacks in real time. These feedbacks are not part of probing traffic, but collected from clients periodically or on-demand. After training the classifier, the service provider can simply input one-way delays, jitters, and loss rates to the MLP and obtain an estimated mean opinion score (MOS) describing end-user experiences. We further note that the inputs are not limited to delay, loss, or jitter, but can be conveniently extended to include any other metrics. Therefore, the resulting learning engine can be applied to QoE management of many other services (e.g., IPTV, VoIP, and software-as-a-service) than IP conference.

The above QoE learning engine, combined with end-to-end probing, allows service providers to real-time monitor users’ experience. Once service quality degradation is determined for certain users, it becomes paramount for service providers to accurately and timely diagnose the root cause and localize the problem down to specific network segments. Towards this end, we respectively present routing topology inference and fault diagnosis schemes in the following two sections.

## IV. ROUTING TOPOLOGY DISCOVERY

Among existing Internet routing topology inference mechanisms, *traceroute*-based techniques are the earliest and most widely used ones [5], [8], [9], [12]. However, these techniques require explicit cooperation of intermediate routers.

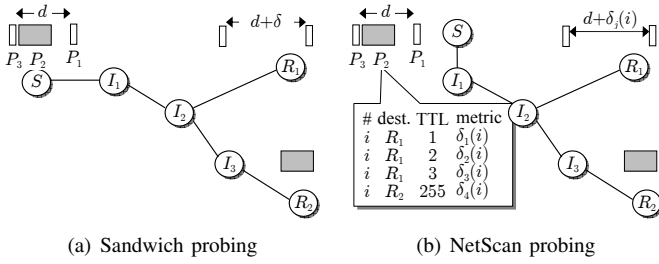


Fig. 1. Illustration of Sandwich and NetScan probing schemes.

For security reasons, more and more routers in today’s Enterprise networks choose not to reveal their identity by either suppressing ICMP responses or using the destination addresses of `traceroute` packets instead of their own as source addresses for outgoing ICMP packets. We call these routers *implicit* or *anonymous* as coined in [21]. In the presence of implicit routers, the conventional `traceroute`-based methods result in routing paths with unidentifiable intermediate routers (which are marked as “\*” by a typical `traceroute` command). Thus, the resulting routing topology may be incomplete or inaccurate [21]. To address this issue, we propose a new topology inference algorithm called NetScan.

### A. Sandwich Probing

Without loss of generality, consider a one-server-two-client system illustrated in Fig. 1. As seen in the figure, a sandwich probe consists of two small packets ( $P_1$  and  $P_3$ ) with packet size  $s_1$  separated by a large packet ( $P_2$ ) with packet size  $s_2$ . The two small packets are sent out by the source with a fixed time interval  $d$ . In a typical scenario shown in Fig. 1(a), source  $S$  first transmits  $P_1$  to the first receiver  $R_1$ . After  $d$  time units, it sends out back-to-back packets  $P_2$  and  $P_3$ , which are destined at  $R_2$  and  $R_1$ , respectively. Then, due to queuing delay experienced by the three packets, the initial time interval  $d$  between  $P_1$  and  $P_3$  is changed by  $\delta$  (i.e., the interval becomes  $d + \delta$ ) at the branching router where paths  $S$ - $R_1$  and  $S$ - $R_2$  diverge. Then, the original sandwich probing scheme reconstructs the logical network topology by applying a penalized maximum likelihood algorithm on  $\delta$  measurements collected at the receivers [6]. Due to limited space, we refer interested readers to [6] for details of sandwich probing.

### B. NetScan

We next present how we modify sandwich probing and leverage TTL decrement to identify the branching node of two paths. This is in spirit similar to Eriksson’s recent work [7], in which TTL decrements are used to infer the length of shared path of two end-hosts. Our scheme is simple. Define  $N_{S,R_1}$  as the number of hops between hosts  $S$  and  $R_1$  and  $m_{S,R_1,R_2} = \min(N_{S,R_1}, N_{S,R_2})$ , which can be used as an upper bound on the length of the shared path between  $S$ - $R_1$  and  $S$ - $R_2$ . Here,  $N_{S,R_1}$  and  $N_{S,R_2}$  can be determined by sending a packet with the default TTL value 255 from  $S$  to  $R_1$  and  $R_2$ , respectively, and then checking the TTL decrements of the packets at the receiving end. Then, our probing algorithm

### Algorithm: BPI

**Input:** Source  $S$  and destinations  $R_1$  and  $R_2$

**Output:** Branching node

Obtain  $N_{S,R_1}$ ,  $N_{S,R_2}$ , and  $m_{S,R_1,R_2} = \min(N_{S,R_1}, N_{S,R_2})$ ;

**for**  $i = 1$  **to**  $K$  **do**

**for**  $j = 1$  **to**  $m_{S,R_1,R_2}$  **do**

    Set TTL of  $P_2$  to  $j$ , send  $P_1$ ,  $P_2$ , and  $P_3$  from  $S$  to  $R_1$ ;

    Measure and calculate  $\delta_i(j)$ ;

  Set TTL of  $P_2$  to 255, send  $P_1$  and  $P_3$  to  $R_1$  and  $P_2$  to  $R_2$ ;

  Measure and calculate  $\delta'(j)$ ;

  Calculate mean  $E[q_j]$  and  $E[q']$ ;

**return**  $\arg_j \min |E[q_j] - E[q']|$ ;

Fig. 2. Branching-point identification algorithm of NetScan.

will initiate  $m_{S,R_1,R_2} + 1$  sandwich probes with different TTL values for the large packet  $P_2$ . Specifically, the large packet  $P_2$  of the  $j$ -th sandwich probe has TTL equal to  $j$  (where  $1 \leq j \leq m_{S,R_1,R_2}$ ) with destination  $R_1$ . For  $P_2$  in the last (i.e., the  $(m_{S,R_1,R_2} + 1)$ -st) sandwich probe, we set its TTL to 255 and destination to  $R_2$ . Consider Fig. 1 for an illustration, where the sandwich probe will be sent four times and TTL values and destinations of  $P_2$  are shown in the table.

In the  $j$ -th round, receiver  $R_1$  measures the variation  $\delta_j(i)$  of the inter-packet interval between the two small packets  $P_1$  and  $P_3$ . Analogously,  $\delta'(i)$  of the last probe reflects the transmission and queuing delay of the sub-path up to the branching point (e.g.,  $I_2$  in Fig. 1(b)). Thus, at the end of the  $K$  rounds, we have  $m_{S,R_1,R_2} + 1$  time series:

$$q_j = (\delta_j(1), \dots, \delta_j(i), \dots, \delta_j(K)), \quad (1)$$

where  $1 \leq j \leq m_{S,R_1,R_2}$ , and

$$q' = (\delta'(1), \dots, \delta'(i), \dots, \delta'(K)). \quad (2)$$

According to [22], the expectation of  $\delta_j$  is approximately equal to the summation of  $s_2/C_i$  and  $\sigma(i)$  along the path  $S$ - $R_1$  up to the  $j$ -th hop. Then assuming  $j$  is the index that minimizes  $|E[\delta_j] - E[\delta']|$ , the branching point will be the  $(j + 1)$ -st node from sender  $S$ . The pseudo-code of this algorithm, Branching Point Identification (BPI), is given in Fig. 2.

## V. ROOT CAUSE ANALYSIS

In QoEScope, end-users’ perceived quality of experience is inferred periodically by the QoE Learning Engine introduced in Section III. Each time the qualities are inferred, QoEScope automatically localizes the most possible network segments that lead to the poor qualities inferred, if any. The problematic links are flagged a value of 1 and these link values are then accumulated and displayed in our control GUI. Detailed description and evaluation of the root cause analysis mechanism used in QoEScope are available in [22].

## VI. EXPERIMENTAL EVALUATION

We implemented the client-side of QoEScope as a Java applet based on two considerations. First, platform-independence

of Java applet enabled Web browser is critical when deployed in customers’ networks with heterogeneous system configurations. Second, its support for sandboxing provides improved security and allows clients to perform probing without installing additional software on their local machines. In cases where agent-based implementation (i.e., software installation on clients’ machines) is desired, our prototype can be easily converted. Using this implementation, we next examine QoEScope’s probing scalability, topology inference accuracy, and QoE learning performance via real experiments conducted on our in-house testbed and Planetlab [17].

### A. Scalability

We perform the scalability test of the QoEScope server to verify how many distributed clients can be supported simultaneously without compromising the accuracy of monitoring. Since, monitoring is done in a centralized architecture, we want to ensure that increase in load does not affect the probing interval on the server side or the measurement data itself by introducing disturbance in the network (e.g., queueing delay in the local router affecting the probing delay). We also want to verify, using a commodity hardware with 100 Mb/s ethernet card, that all probing clients are supported and whether there is any requirement on the IP server bandwidth and how increased probing affects it (or affects the underlying IP conference traffic). Our testing setup is explained below.

The QoEScope server was collocated with the IP conference server and run on a Linux box with a 2.4 GHz CPU, 4 GB RAM, and 100 Mb/s ethernet card. We run the instances of probing clients from 4 different client machines, two of which are Windows XP (3 GHz CPU and 1 GB RAM) and the other two are Windows Vista (2.33 GHz CPU and 3 GB RAM). Multiple probing client processes were created in each client machine. Each probing client was launched with a one-second interval. Fig. 3(a) plots CPU, memory, and network utilization as functions of the number of clients. As seen from the figures, these metrics increase linearly with the number of probing clients, which is expected. In addition, from the measurement traces we verified that, probing packet intervals on the server were not affected while the CPU utilization was less than 99%. This implies that the probing accuracy was not compromised on the server side by the client-generated probing traffic. We also verified that there are no packet loss during the entire experiment duration.

### B. Topology Discovery

In this subsection, we evaluate performance of QoEScope’s topology discovery scheme (NetScan) in Planetlab. We choose 20 Planetlab nodes, 15 of which are US nodes and 5 are overseas. We conduct 100 rounds of NetScan probing and identification processes, in each of which we randomly choose one node as the sender and two nodes as receivers. We also run `traceroute` from the sender to receivers to obtain the ground-truth routing topology. To emulate anonymous routers, we randomly choose a set of routers and manually replace their IP addresses in the `traceroute` results with \*. We plot the

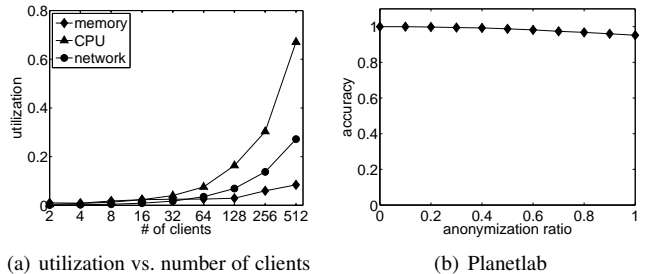


Fig. 3. (a) Scalability test of QoEScope on in-house testbed with one server and four physical client machines; (b) Accuracy of QoEScope’s topology inference mechanism in Planetlab.

measurement results in Fig. 3(b), from which we can see that QoEScope is very robust to anonymous routers. Particularly, even with 100% anonymous routers, NetScan still can achieve 95% accuracy. This makes QoEScope highly appealing for heterogeneous enterprise networks.

### C. QoE Learning Engine

We also study the performance of QoEScope’s QoE learning engine. We use the same testbed as in the scalability test presented in Section VI-A. In this testbed, traffic of each client passes through a separate Dummynet pipe, which allows us to independently configure delay and loss rate of individual clients. During the course of the measurements, we also concurrently launched our IP video conference system to emulate realistic background traffic. We conducted two sets of experiments, in both of which delay and packet loss of each client are randomly and dynamically updated. These two data sets include 300 and 200 measurement samples, respectively. We use the first data set as the labeled data for training the classifier and the second set as the test set. In particular, we are interested in studying the impact of training data size on prediction accuracy on the test set.

The experiment results are given in Table I, in which we calculate the correlation coefficient ( $CC$ ), mean absolute error (MAE), root mean square error (RMSE), relative absolute error (RAE), and root relative square error (RRSE) between the actual  $R$ -scores and those predicted by QoEScope using a training set of different sizes. From the table we can see that as the size of training set increases, prediction accuracy quickly approaches the optimal case, which is obtained by using the second data set as both training and test sets. Specifically, when the training set contains 256 samples, performance of the classifier almost achieves optimality. This demonstrates that the MLP classifier utilized by QoEScope is very accurate to describe the relationship between QoS and QoE metrics and requires a small set of labeled data for initial training.

## VII. RELATED WORK

Several works are especially related to ours. [1], [4] tried to infer root causes of failing network services by discovering dependency graphs among network applications and network elements. Our work focuses more on locating problematic network segments that causes poor user perceived experience.

TABLE I  
QOE LEARNING ENGINE: ACCURACY VS. SIZE OF TRAINING SET.

	Size of training set								Optimum
	2	4	8	16	32	64	128	256	
<i>CC</i>	0.73	0.98	0.99	0.99	0.99	0.99	0.99	1.00	1.00
<i>MAE</i>	14.22	10.87	4.19	3.45	1.51	0.96	0.91	0.33	0.32
<i>RMSE</i>	15.66	12.45	5.42	4.79	3.29	1.49	1.25	0.45	0.37
<i>RAE (%)</i>	49.09	38.79	16.16	14.03	6.01	2.97	1.86	1.34	1.19
<i>RRSE (%)</i>	50.16	41.79	20.16	19.04	12.74	4.13	2.32	1.90	1.77

Our work is also related to network performance inference problems such as [16], where network link losses are inferred from server traces through various approaches.

There is also a large body of literature in network topology inference, which is in spirit similar to NetScan. Specifically, among existing Internet routing topology inference mechanisms, `traceroute`-based techniques are the earliest and most widely used ones [5], [8], [9], [12]. However, as discussed in Section II, these schemes may not work well in the presence of implicit routers. To overcome the drawbacks of `traceroute`-based schemes, many new topology-inference techniques have been proposed. In particular, Coates *et al.* used the *Sandwich Probing* scheme to infer the routing tree topology for one sender and multiple receivers [6]. Later on in [19], Rabbat *et al.* showed that the multiple source, multiple destination logical topology inference problem can be reduced to the two source, two destination case and then proposed a method for testing whether links are shared in the two source, two destination case. In [14], Mao *et al.* used historical data to infer the local structure based on nonnegative matrix factorization and then diagnosed faults using the inferred local topology. In [15], the logical routing tree is constructed based on additive metrics (e.g., loss rate, utilization, and delay). In [2], metric-induced network topologies are constructed using end-to-end measurements.

## VIII. CONCLUSIONS

In this paper, we presented design and experimentation of QoEScope for the management of enterprise IP service systems with a client-server structure (e.g., IP conference and IPTV). QoEScope integrates scalable end-to-end probing, adaptive learning of user-perceived service quality, accurate and robust topology inference, and automatic root cause analysis into a single QoE management solution. As discussed in the paper, although presented in the context of IP conference systems, QoEScope can be easily extended to other IP service systems by customizing the learning component. Performance of QoEScope has also been evaluated using real experiments conducted in local testbed and Planetlab. Our future work involves field test of QoEScope in a production IP service system, enhancement of probing scalability, and design of more robust and accurate root cause analysis schemes.

## REFERENCES

[1] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D. A. Maltz, and M. Zhang, "Towards Highly Reliable Enterprise Network Services via Inference of Multi-level Dependencies," in *Proc. ACM SIGCOMM*, Aug. 2007, pp. 13–24.

[2] A. Bestavros, J. W. Byers, and K. A. Harfoush, "Inference and Labeling of Metric-Induced Network Topologies," *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, no. 11, pp. 1053–1065, Nov. 2005.

[3] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu, "Network tomography: recent developments," *Statistical Science*, vol. 19, pp. 499–517, 2004.

[4] X. Chen, M. Zhang, Z. M. Mao, and P. Bahl, "Automating Network Application Dependency Discovery: Experiences, Limitations, and New Solutions," in *Proc. USENIX OSDI*, Oct. 2008.

[5] B. Cheswick, H. Burch, and S. Branigan, "Mapping and Visualizing the Internet," in *Proc. USENIX Annual Technical Conference*, Dec. 2000, pp. 1–1.

[6] M. Coates, R. Castro, and R. Nowak, "Maximum Likelihood Network Topology Identification from Edge-Based Unicast Measurements," in *Proc. ACM SIGMETRICS*, Jun. 2002, pp. 11–20.

[7] B. Eriksson, P. Barford, and R. Nowak, "Network Discovery from Passive Measurements," in *Proc. ACM SIGCOMM*, Aug. 2008, pp. 291–302.

[8] CAIDA: Cooperative Association for Internet Data Analysis. [Online]. Available: <http://www.caida.org/tools/>. Accessed on Aug. 10, 2008.

[9] R. Govindan and H. Tangmunarunkit, "Heuristics for Internet Map Discovery," in *Proc. IEEE INFOCOM*, Mar. 2000, pp. 1371–1380.

[10] ITU-T Recommendation G.107, "The E-Model, a Computational Model for Use in Transmission Planning," Mar. 2005.

[11] N. Jariyakul, "An Optimized Clustering and Selective Probing Framework to Support Internet Quality-of-Service Routing," *Simulation*, vol. 82, no. 5, pp. 311–330, 2006.

[12] X. Jin, W. Tu, and S.-H. Chan, "Traceroute-Based Topology Inference without Network Coordinate Estimation," in *Proc. IEEE ICC*, May. 2008, pp. 1615–1619.

[13] E. Kohler, M. Handley, and S. Floyd, "Designing DCCP: Congestion Control Without Reliability," in *Proc. ACM SIGCOMM*, Sep. 2006, pp. 27–38.

[14] Y. Mao, H. Jamjoom, S. Tao, and J. M. Smith, "NetworkMD: Topology Inference and Failure Diagnosis in the Last Mile," in *Proc. ACM SIGCOMM IMC*, Oct. 2007, pp. 189–201.

[15] J. Ni, H. Xie, S. Tatikonda, and Y. R. Yang, "Network Routing Topology Inference from End-to-End Measurements," in *Proc. IEEE INFOCOM*, Apr. 2008, pp. 36–40.

[16] V. Padmanabhan, L. Qiu, and H. Wang, "Server-based inference of Internet link lossiness," in *Proc. IEEE INFOCOM*, Apr. 2003, pp. 145–155.

[17] Planetlab. [Online]. Available: <https://www.planet-lab.org/>. Accessed on Feb. 1, 2009.

[18] T. Qiu, J. Ni, H. Wang, N. Hua, Y. R. Yang, and J. Xu, "Packet Doppler: Network Monitoring using Packet Shift Detection," in *Proc. ACM CoNext*, Dec. 2008.

[19] M. G. Rabbat, M. J. Coates, and R. D. Nowak, "Multiple-Source Internet Tomography," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 12, pp. 2221–2234, Dec. 2006.

[20] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," *IETF RFC 3550*, Jul. 2003.

[21] B. Yao, R. Viswanathan, F. Chang, and D. Waddington, "Topology Inference in the Presence of Anonymous Routers," in *Proc. IEEE INFOCOM*, Apr. 2003, pp. 353–363.

[22] Y. Zhang, Y. Ru, and G. Jiang, "NetScan: Inferring Physical Network Topology from End-to-End Measurements," NEC Labs America, Inc., Tech. Rep. 2009-L030, Jan 2009. [Online]. Available: <http://www.nec-labs.com/~yueping/netscan.pdf>.