

QoSaaS: Quality of Service as a Service

Ye Wang
Yale University

Cheng Huang
Microsoft Research

Jin Li
Microsoft Research

Philip A. Chou
Microsoft Research

Y. Richard Yang
Yale University

Abstract

QoSaaS is a new framework that provides QoS information portals for enterprises running real-time applications. By aggregating measurements from both end clients and application servers, QoSaaS derives the quality information of individual system components and enables system operators to discover and localize quality problems; end-users to be hinted with expected experience; and the applications to make informed adaptations. Using a large-scale commercial VoIP system as a target application, we report preliminary experiences of developing such service. We also discuss remaining challenges and outline potential directions to address them.

1 Introduction

Despite recent large-scale deployments, enterprise real-time applications such as VoIP and video conferencing still face significant Quality of Service problems. For example, our own measurements collected from a commercial enterprise VoIP system show that as many as 26% of VoIP sessions experienced poor quality (Section 2).

System operators would like to localize the problems in order to apply fixes such as hardware upgrade, software patch, and infrastructure provision. End-users could benefit from visual hints about expected experience before each session. The real-time applications themselves prefer to make their adaptations informed and effective, such as how to adjust redundancy level, adapt coding rate or route through alternative paths (*e.g.*, [2, 13, 20, 24]). For all these interests, it appears beneficial to monitor the quality of the individual system components and make the information available *in real-time*.

In this paper, we propose a new portal service named Quality of Service as a Service (QoSaaS, or simply QoS Service), as an essential supporting service for enterprises running real-time applications. QoSaaS aggregates quality measurements from end clients and application servers, derives the quality information of individual system components, and provides an informational service to system operators, end-users, and the applications, for diverse purposes including offline diagnosis, online quality visualization, and application adaptations.

Any single session of the real-time applications may involve a large number of system components, where many components (such as wireless links, layer 2 switches, etc.) are beyond the control of the applications. Problems in a particular component may mani-

fest as quality degradation in end-to-end sessions, but observed end-to-end quality degradation could be attributed to any of the inline components. Moreover, individual system components only have a limited view of an entire system. Therefore, a key challenge of QoSaaS lies in developing an inference engine that can efficiently and accurately derive the quality information of individual system components from the collected measurements.

In this paper, using the large-scale commercial VoIP system as a target application, we report preliminary experience of developing the QoS Service. We focus on packet loss rate as the quality metric and employ a Maximum Likelihood Estimation-based inference algorithm. We present the inferencing results for individual components of the system and examine a number of identified problematic components in detail.

We then point out the limitations of the preceding approach, which include: *i*) it does *not* identify problems occurring over small time scales; *ii*) it does *not* readily generalize to other quality metrics, such as delay jitter; *iii*) it does *not* distinguish and therefore would be confused by impairments occurring locally on end clients. To address these limitations, we outline a few potential solutions as future directions.

The rest of the paper is organized as follows. We first motivate the QoS Service with our measurements of the large-scale commercial audio/video conferencing system (Section 2). Section 3 presents the overall design of QoSaaS. In Section 4, we describe the QoS inferencing problem and present our current solution with preliminary results. Section 5 discusses the remaining challenges and points out potential directions. Related works are reviewed in Section 6.

2 Motivation

Our work is motivated by quality problems encountered in real-world operation of Microsoft Lync, a commercial Unified Communications solution that has been deployed within Microsoft and many other global enterprises. A key component of Lync is a VoIP system that completely replaces traditional telephony service and supports all daily communications in an enterprise.

2.1 VoIP Quality of the Lync System

Lync is a large-scale and complex system, as it supports all of the communication scenarios required by a variety of business needs. The simplest scenario is a direct call between two end-points within the same enterprise

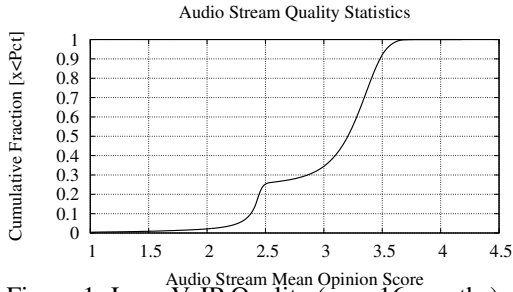


Figure 1: Lync VoIP Quality (over 16 months).

network. In this case, a direct peer-to-peer connection is established for the VoIP session. Other scenarios are more complex and often involve media servers. Lync deploys three types of media servers: 1) a media conferencing server serving as a mixer in multi-party audio and video conferencing; 2) a relay server connecting end-points from the public Internet to those within the enterprise; and 3) a mediation gateway bridging VoIP sessions with PSTN.

Here is an example involving both a relay server and a mediation gateway. Say, an enterprise user in US, while working at home, makes a call to a PSTN phone in Europe. In this case, the VoIP session first goes through a tunnel from the user’s home to a relay server in US. Next, the VoIP session is routed to Europe within the global network infrastructure of the enterprise. Finally, it is routed to a mediation gateway in Europe so as to bridge with PSTN and eventually reach the PSTN phone.

The deployment scale of the Lync system is summarized in Table 1. Ensuring QoS across all scenarios in

Deployed Countries	85
Users (enterprise plus external)	~ 7,000,000
Daily (weekday) Audio Sessions	~ 500,000
Used Global Data Centers	3
Running Media Servers	~ 150

Table 1: Deployment Scale of the Lync System.

such scale is extremely challenging. Indeed, both user feedback and collected statistics indicate that the system suffers from significant quality problems. To quantify the quality of experience, we plot the Mean Opinion Score (MOS) [11] reported by all of the VoIP sessions over 16 months in Figure 1. We observe that 26% of the VoIP sessions experience MOS lower than 2.5 – a typical threshold for acceptable quality.

2.2 Challenges of Ensuring Quality

An individual VoIP session may involve a large number of system components, and quality degradation could be attributed to any of the inline components. For instance, a single VoIP session might traverse a large set of network components (*e.g.*, load balancers, routers, and network links) and media servers (*e.g.*, mediation gateways, relay servers, and conferencing servers). Any component along the path may cause quality degradation: end-points might capture poor quality audio due to device is-

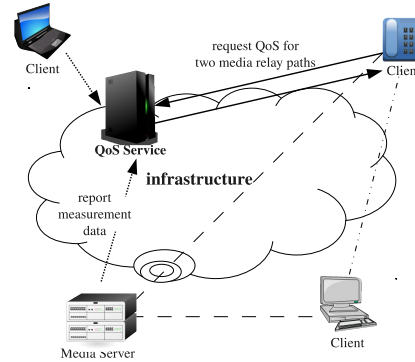


Figure 2: QoSaaS Overview.

sues; VoIP packets might be dropped in wireless links and congested access gateways or network connections; media servers might delay packets due to overload, mis-configuration, and/or software bugs. On the other hand, individual system components (*e.g.*, end-points, media servers, load balancers, and routers) only have a limited view of the overall system.

A consequence is the lacking, in the current enterprise network, of information about how each system component affects VoIP quality end-to-end. Indeed, the aim of the proposed QoS Service is to fill such a void.

3 QoSaaS Design

The goal of the QoS service is to provide QoS information so as to substantially improve the user experience of real-time applications in enterprises. In a nutshell, the QoS service aggregates end-to-end quality measurements from end-points and media servers, derives the QoS information of individual system components, and makes QoS predictions on arbitrary end-to-end paths.

3.1 Architecture Overview

Figure 2 illustrates the basic components and message flows of the QoS service. At the center of the service is a logical QoS server. The QoS server is bootstrapped with information about network topology and the deployment of media servers. The end-points and media servers collect QoS measurements about individual VoIP sessions and report the measurements to the QoS server. The QoS server employs an inference engine and derives the QoS degradation caused by individual system components. System operators can query the QoS server for the quality estimation of arbitrary system component. The end-points can query the QoS server for the quality estimation of arbitrary end-to-end path.

The QoS service consists of the following three major components:

- *Measurement Aggregation*: The end-points and media servers collect QoS measurements of individual VoIP sessions. The measurements are reported to the QoS server, which aggregates the information in specific ways that we will elaborate in later sections.
- *Component QoS Inference*: This component is the core of the QoS service. Given the measurements, the

inference engine infers the QoS degradation at each individual system component in the VoIP system.

- *QoS Prediction*: Given recent and historical inference results, this component predicts the QoS along an arbitrary path consisting of any network components and media servers.

3.2 QoS Service Usage

QoSaaS can be used by both the system operators and the real-time network applications.

Because the QoS service infers QoS information of individual system components, the operators can query the service and discover problematic network segments and media servers. For example, if the QoS service reports consistently high packet loss rate at a particular subnet, the operators can focus on the network gateway of the subnet to further diagnose the problem and apply fixes.

For real-time applications such as multimedia communication, whenever there is a quality problem due to system components, an end-point can query the QoS service for the quality estimation of multiple alternative paths. It can then migrate a VoIP session to the path offering the best performance. For instance, if the default mediation server is problematic, the VoIP session can be routed to a different city so as to bridge with PSTN through another mediation server. If the direct path between a caller and a callee experiences a problem, a detour path can be established through a conferencing server, or through other end-points if they are able to serve as relay points.

Even when no alternative path is available or even alternative paths do *not* improve quality, the QoS estimation of the existing path can still be useful. The end-points can display the QoS information to the end-users so that they can form proper expectation about the VoIP session. This is analogous to signal bars displayed on cell phones.

4 Preliminary Experiences

The core of the QoS service is a quality inference engine. The inference engine aggregates end-to-end measurements and derives quality estimation of individual system components, including network segments and media servers. In this section, we describe our preliminary experience developing the inference engine employing a Maximum Likelihood Estimation-based approach.

4.1 System Model

We model the VoIP system as a *directed* graph G . The nodes in the graph model end-points, media servers, and physical locations (such as cities). The edges in the graph model network segments connecting the nodes. As illustrated in Figure 3, the media server in Dublin is a node, and city Beijing is also a node, while subnet $x.x.130.0/23$ connecting client a to city Redmond is an edge.

Both nodes and edges can cause quality degradation.

Therefore, they are regarded as media transmission *entities*. The edges are directed as degradation along different directions is typically asymmetric.

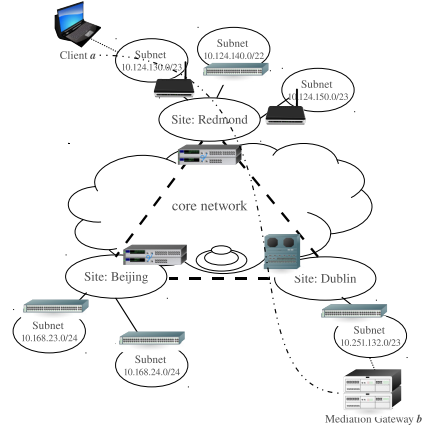


Figure 3: System Model (with media line $M_{a \rightarrow b}$).

The system model abstracts away the physical network topology. For instance, there might be more than one physical routers in a city, or more than one physical network links between two cities. The advantage of such abstraction is that it allows the inference to be refined hierarchically. Once a problem is localized to a node or an edge, detailed diagnosis tools [1, 3, 14] can be employed to identify root causes.

A *media line* defines a directed path in the graph linking multiple entities. As illustrated in Figure 3, a media line, denoted as $M_{a \rightarrow b}$, connects client a to mediation gateway b , with four other entities in between: subnet $x.x.130.0/23$, city Redmond, edge between Redmond and Dublin, and subnet $x.x.132.0/23$.

A two-party direct call consists of a pair of media lines, one from the caller to the callee and the other in the reverse direction. A multi-party conference, on the other hand, involves a conferencing server to mix audio and therefore consists of more media lines. In particular, there is one pair of media line between each participating end-point and the conferencing server. The destination node of every media line collects and reports the QoS of the media line to the QoS service.

4.2 Maximum Likelihood Based Inference

For every VoIP session, the QoSaaS service collects QoS measurements from each media line. The inference engine then use these measurements to derive estimation of individual entities. In this section, we focus on an example QoS metric – packet loss rate – and develop a maximum likelihood based inference algorithm.

Assume that media line $M_{a \rightarrow b}$ consists of k entities (e_1, e_2, \dots, e_k). Denote the packet loss rate of entity e_i as p_i . Assume that packet loss is independent. Then, the aggregated packet loss rate of the media line is $p_M = 1 - \prod_{i=1}^k (1 - p_i)$.

Now, consider all the packets transmitted from a to b .

Let n_M denote the number of transmitted packets and m_M the number of received. Then, the number of lost packets is $(n_M - m_M)$. Therefore, the likelihood of such events (n_M transmitted and m_M received) is

$$L_M(n_M, m_M) = p_M^{(n_M - m_M)} (1 - p_M)^{m_M}. \quad (1)$$

Here, n_M and m_M are two metrics easy to collect. Alternatively, n_M can be estimated readily from the duration of the VoIP session, and m_M can be calculated based on the average end-to-end loss rate.

Given the likelihood of the events on individual media lines, the likelihood over the entire system, denoted as L , is the product of the individual likelihoods. Then, maximum likelihood estimation assigns packet loss rates to individual entities so as to maximize the system-wide likelihood, as follows:

$$\{p_i\} = \arg \max_{\{p_i\}} \{L = \prod_{\{M\}} L_M(n_M, m_M)\}. \quad (2)$$

Note that the maximum likelihood estimation naturally takes into account the duration of VoIP sessions – longer sessions have more influence than shorter ones – while simple linear models [4, 7, 27] completely ignore the session duration.

4.3 Inferencing Results

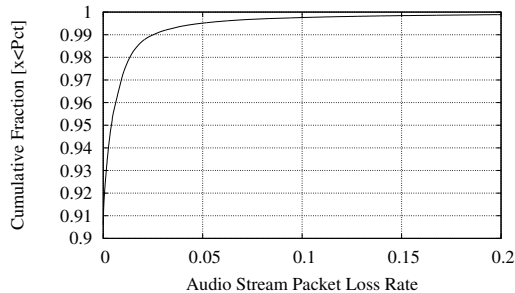


Figure 4: Packet Loss Rate of VoIP Audio Streams.

Using measurements collected from a typical work week (from July 20 to July 24), we now present preliminary inferencing results.

All the measurements are grouped by 15-minute slots. Within each 15-minute slot, the packet loss rate of an entity is assumed to be independent and constant. Typically, we observe around 150 entities (out of about 4000) relaying about 5,000 VoIP audio streams. The concentrated work load on a small fraction of entities is mainly due to time zone difference. For instance, when the network and the media servers in North America are heavily utilized, those in Asia are mostly idle.

Figure 4 plots the packet loss rate of all the audio streams. Clearly, a small fraction of the VoIP sessions suffer from significant QoS degradation.

4.3.1 Overview

The inferencing results are summarized in the heat map shown in Figure 5. The heat map encodes the packet

loss rates of 350 entities into colored dots, one per 15-minute slot. The higher the packet loss rate, the brighter the color is. Note that the heat map only includes entities with non-zero packet loss rate in at least one slot, excluding around 3600 entities that never drop packets.

From the heat map, we can clearly observe: *i)* wireless subnets incur more packet losses than wired ones; *ii)* a few wired subnets are plagued by frequent losses; *iii)* certain site-to-site network segments also experience losses once in a while; *iv)* media servers are healthy most of the time.

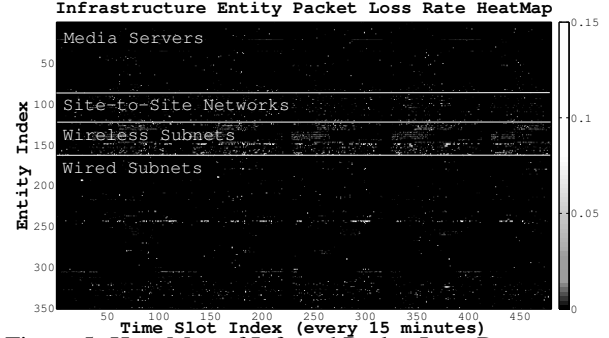


Figure 5: Heat Map of Inferred Packet Loss Rate.

4.3.2 Case Studies

Next, we illustrate a few concrete case studies.

Issue 1: Wireless AP Gateway Poor Performance In the heat map, we observe one particular wireless subnet $x.x.158.64/27$ (entity 149) constantly experiences high packet loss rate. The subnet maps to a wireless AP gateway installed in an office building in Charlotte, NC. The inferred packet loss rate of the subnet is plotted in Figure 6. The results indicate that, on average, 4.6% of VoIP packets can be dropped, and the packet loss rate can be as high as 20% during busy hours.

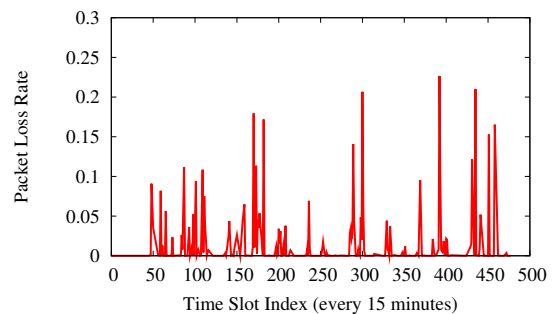


Figure 6: Packet Loss Rate (Wireless Subnet).

Issue 2: Long Distance Network Degradation Figure 7 plots the inferred packet loss rate on long distance network connections between Hyderabad, India and Redmond, US. The connection from Hyderabad to Redmond corresponds to entity 98, while that from Redmond to Hyderabad maps to entity 114 in the heat map.

Issue 3: Media Relay Server Bug Users in Redmond complained about poor VoIP quality when calling exter-

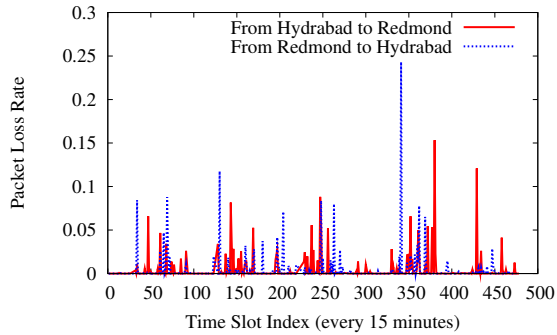


Figure 7: Packet Loss Rate (Hyderabad – Redmond).

nal users. System administrators added one media relay server `x.x.141.95` to alleviate the load on existing ones. Unfortunately, the problem was not solved and audio quality remained poor.

We plot the inferred packet loss rate of related media relay servers in Figure 8. The newly added media relay server appears to incur significantly higher packet loss rate, as shown by the fine-dashed curve starting from time slot 118, than a normal media server `x.x.141.81`. On the other hand, one of the existing media relay servers `x.x.141.92` also experience high packet loss.

The problem was eventually attributed to a software bug in the media server stack, and both server `x.x.141.92` and `x.x.141.95` happened to operate the same version of stack.

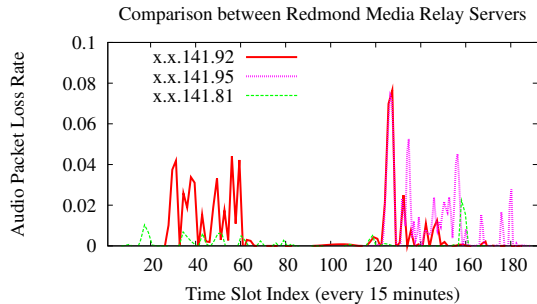


Figure 8: Packet Loss Rate (Relay Servers in Redmond).

5 Challenges and Directions

Section 4 has shown that statistics collected over VoIP sessions (*e.g.*, average packet loss rate) is helpful in inferring the QoS of individual components of the system. However, coarse-grained statistics can discover only simple problems that persistently manifest over long time scale. The examples shown in Section 4 all bear this nature. Many problems in real systems are much more stealthy, where coarse-grained statistics often becomes insufficient.

In this section, through analyzing real problems discovered in production systems, we elaborate on a few key challenges and point out potential directions so that the QoSaaS framework can be generalized to easily deal with new problems in future.

5.1 Time Resolution

Let’s examine a problem related to time scale. A media server runs a dedicated process to collect a large number of performance counters. On Windows Server 2003, this process wakes up every 15 minutes and blocks one of the CPU cores completely for 200 ms. As a result, VoIP sessions using the blocked core are affected and experience a 200 ms jitter. This problem is difficult to discover. The 200 ms jitter occurs only once every 15 minutes. It can *not* be observed in average jitter over an entire VoIP session. In addition, this problem only affects the VoIP sessions using the blocked core, but *not* other sessions using unblocked cores on the same server.

Packet-level traces can help the discovery of such problem occurring at small time-scale. A packet-level trace can be compactly represented, *e.g.*, the trace of a 5-minute VoIP session with audio rate at 50 packets per second is about 60 KB [19]. Even so, collecting all the packet-level traces from *every* end-point is infeasible and unnecessary, due to redundant information carried by the traces. Therefore, it is desirable to collect only sufficient traces for inference and no more. To this end, one approach is to instrument every end-point to record a packet-level trace for each VoIP session and store the trace locally. An end-point only uploads its traces upon the request from the QoS server.

5.2 Jitter Inference

Given packet-level traces, packet losses at small time-scale can be inferred. Jitter, however, needs to be derived in a different way. Averaging even over small time-scale bears the risk of diluting useful information. In the above example, the single jitter of 200 ms equates to an average jitter of only 4 ms per second, if the audio rate is 50 packets per second. Therefore, it is more effective to first derive per packet jitter value at individual system components and then aggregate statistics.

Given end-to-end jitter measurements, jitter inference tries to find the most probable assignment of jitter values to individual network components over various time instances. Similar to the loss rate estimation, the most probable solution tends to concentrate assignments to only a few components. Yet, how to derive such assignments appears quite different. One potential approach is to explore time dependency between jitter values on each component. Intuitively, if a component incurs a non-zero jitter at time t , then it is more likely to incur another non-zero jitter at time $t + 1$ than another component that appears perfect recently.

5.3 Local Impairment

All problems are *not* caused by the network components. Often times they are due to local impairments occurring on the end-points. Consider another real problem. This problem is observed on a slow machine with a single 1.8 GHz AMD core CPU, a typical netbook configuration.

Severe audio jitters are observed, which are caused by a time critical system thread hogging CPU for 80 ms from time to time. The threads of the VoIP application run at the same priority as the critical thread and thus experience glitches of 80 ms in audio processing.

Obviously, packet-level traces collected on this machine will contain many jitters. The inference based on the traces would have falsely assign jitters to components in network. To reduce the false alarm, it is crucial to distinguish local impairments from those caused by the infrastructure.

One solution to make such distinction is to explore the bi-direction nature of VoIP sessions. There are simultaneous outgoing and incoming packets from the participating end-points. Outgoing packets are only affected by the local end-point and have nothing to do with the network, while incoming packets are affected by the local end-point, the network and/or the remote end-point. By comparing the traces between the outgoing and incoming packets, it is possible to distinguish local impairments from network QoS degradation. Note that modern VoIP systems employ voice activity detection, so that audio is suppressed and no outgoing packets are sent during silence periods. We may add virtual outgoing packets to handle such artificially irregular traces.

6 Related Work

Quality of Service has long been recognized as an important topic in network services. There are many previous studies on QoS related infrastructure support, resource reservation and allocation, as well as system parameter tuning (e.g., [6, 12, 25]).

Recently, large-scale multimedia communication systems have been deployed in enterprises and the public Internet. Consequently, many QoS problems are observed [18] and analyzed [8]. Also, QoS assessment techniques are developed to better evaluate real systems [11, 23].

Many network diagnostic tools can be used to identify system performance problems [3, 14]. Specific tools for troubleshooting multimedia systems are also introduced, for example, the Giza tool set for IPTV system diagnosis [17]. Different from these systems, the scope of QoSaaS is much broader and beyond localizing failure points in the systems.

There are other proposed services that end-users can query for coarse-grained network delay, loss rate, cost, and other path properties (e.g., P4P and ALTO service [22, 26], iPlane [15, 16], and Sequoia [21]). QoSaaS differs from these services as it aims at improving the quality of real-time multimedia communication systems. In such systems, many problems occur only in small time resolution. Therefore, the QoS service is required to provide quality estimation at much finer granularity. In addition, problems could be due to local impairments. So, the QoS service also needs to distinguish the local im-

pairments from infrastructure problems.

One of the core components of QoSaaS is the inference engine. There are a lot of related studies in the network tomography area [4, 5, 7, 9, 10, 27]. Different from the linear topology model and sampled measurements in these studies, we employ a Maximum Likelihood based inference algorithm that derives quality estimates using aggregated measurements from a large number of VoIP sessions.

References

- [1] M. K. Aguilera, J. C. Mogul, J. L. Wiener, P. Reynolds, and A. Muthicharoen. Performance debugging for distributed systems of black boxes. In *ACM SOSP*, 2003.
- [2] Y. Amir, C. Danilov, S. Goose, D. Hedqvist, and A. Terzis. 1-800-OVERLAYS: using overlay networks to improve VoIP quality. In *NOSS-DAV*, 2005.
- [3] 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. *ACM CCR*, 37(4), 2007.
- [4] A. Bestavros, J. W. Byers, and K. A. Harfoush. Inference and labeling of metric-induced network topologies. *IEEE Trans. Para. Dist. Syst.*, 16(11), 2005.
- [5] T. Bu, N. Duffield, F. L. Presti, and D. Towsley. Network tomography on general topologies. *ACM SIGMETRICS Perf. Eval. Review*, 30(1), 2002.
- [6] A. Campbell, G. Coulson, and D. Hutchison. A quality of service architecture. *ACM CCR*, 24(2), 1994.
- [7] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu. Network tomography: Recent developments. *Statistical Science*, 19(3), 2004.
- [8] A. Clark. VoIP performance management. In *Internet Telephony Conference*, 2005.
- [9] M. Coates, R. Castro, R. Nowak, M. Gadhiok, R. King, and Y. Tsang. Maximum likelihood network topology identification from edge-based unicast measurements. In *ACM SIGMETRICS*, 2002.
- [10] M. J. Coates and R. D. Nowak. iplane nano: path prediction for peer-to-peer applications. In *ITC Conference on IP Traffic, Modeling and Management*, 2001.
- [11] R. G. Cole and J. H. Rosenbluth. Voice over IP performance monitoring. *ACM CCR*, 31(2), 2001.
- [12] J. Gozdecki, A. Jajszczyk, and R. Stankiewicz. Quality of Service terminology in IP networks. *IEEE Communications Magazine*, 2003.
- [13] M. Jain and C. Dovrolis. Path selection using available bandwidth estimation in overlay-based video streaming. *Computer Networks*, 52(12), 2008.
- [14] S. Kandula, R. Mahajan, P. Verkaik, S. Agarwal, J. Padhye, and P. Bahl. Detailed diagnosis in enterprise networks. In *ACM SIGCOMM*, 2009.
- [15] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane: an information plane for distributed services. In *USENIX OSDI*, 2006.
- [16] H. V. Madhyastha, E. Katz-Bassett, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane Nano: path prediction for peer-to-peer applications. In *USENIX NSDI*, 2009.
- [17] A. A. Mahimkar, Z. Ge, A. Shaikh, J. Wang, J. Yates, Y. Zhang, and Q. Zhao. Towards automated performance diagnosis in a large IPTV network. In *ACM SIGCOMM*, 2009.
- [18] A. P. Markopoulou, F. A. Tobagi, and M. J. Karam. Assessing the quality of voice communications over Internet backbones. *IEEE/ACM Trans. Networks*, 11(5), 2003.
- [19] A. Mondal, R. Cutler, C. Huang, J. Li, and A. Kuzmanovic. SureCall: Towards glitch-free real-time audio/video conferencing. In *IWQoS*, 2010.
- [20] H. Radha, M. van der Schaar, and Y. Chen. The MPEG-4 fine-grained scalable video coding method for multimedia streaming over IP. *IEEE Trans. Multimedia*, 3:53–68, 2001.
- [21] V. Ramasubramanian, D. Malkhi, F. Kuhn, M. Balakrishnan, A. Gupta, and A. Akella. On the treeness of Internet latency and bandwidth. In *ACM SIGMETRICS*, 2009.
- [22] J. Seedorf and E. Burger. *Application-Layer Traffic Optimization (ALTO) Problem Statement, RFC 5693*, October 2009.
- [23] L. Sun. Voice quality prediction models and their application in VoIP networks. *IEEE Trans. Multimedia*, 2006.
- [24] S. Tao, K. Xu, A. Estepa, T. Fei, L. Gao, R. Guerin, J. Kurose, D. Towsley, and Z.-L. Zhang. Improving VoIP quality through path switching. In *IEEE INFOCOM*, volume 4, 2005.
- [25] A. Vogel, B. Kerhervé, G. v. Bochmann, and J. Gecsei. Distributed multimedia applications and quality of service: a survey. In *CASCON*, 1994.
- [26] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. G. Liu, and A. Silberschatz. P4P: provider portal for applications. In *ACM SIGCOMM*, 2008.
- [27] Y. Zhao, Y. Chen, and D. Bindel. Towards unbiased end-to-end network diagnosis. *IEEE/ACM Trans. Networks*, 17(6), 2009.