Statistical Aspects of the Analysis of Data Networks

Lorraine DENBY, James M. LANDWEHR, Colin L. MALLOWS, Jean MELOCHE, John TUCK

Data Analysis Research
Avaya Labs
Basking Ridge, NJ 07920

Bowei XI

Department of Statistics
Purdue University
West Lafayette, IN 47907

George MICHAILIDIS, Vijayan N. NAIR

Department of Statistics
University of Michigan
Ann Arbor, MI 48109

Assessing and monitoring the performance of computer and communications networks is an important problem for network engineers. A considerable amount of work has been done on tools and techniques for data collection, modeling, and analysis within the network research community. This article presents an overview of the engineering problems and statistical issues, describes recent research developments, and summarizes ongoing work and areas for further research. Although there are many interesting issues related to network analysis, our focus here is on estimating and monitoring network quality-of-service parameters. We discuss methods for estimating edge-level parameters from end-to-end path-level measurements, an important engineering problem that raises interesting statistical modeling issues. Other topics include network monitoring, network visualization, and discovery of network topology. Data from a corporate network are used to illustrate the problems and techniques. As in any overview, the discussion is likely to be slanted toward the authors' own research interests.

KEY WORDS: Internet telephony; Network monitoring; Network tomography; Network visualization; Probe packet; Traceroute data; Voice-over-IP.

1. INTRODUCTION

Modern computer and communications networks are complex, dynamic systems with millions of users and a broad range of applications. Business customers as well as individual users expect consistently high levels of quality, especially when they are running complex applications, such as telephony and video. Thus one must continuously collect, model, and analyze relevant data to assess and monitor network performance. There are many interesting statistical issues and challenges involved in doing this. The present article provides background on the engineering problems and an overview of the statistical issues, recent developments, and ongoing research. The article includes a review of existing results and a discussion of some new results. Predictably, however, the discussion emphasizes our own interests.

Internet protocol (IP) telephony (also known as voice-over-internet protocol, or VoIP) is an important application and provides a basis for much of our discussion in this article. IP telephony typically involves a pair of IP phones that send streams of packets carrying voice traffic. At the sending phone, the packets are sent with regularity (e.g., every 20 msec), and each packet contains a segment of voice. At the receiving phone, the packets do not arrive with the same regularity because of unpredictable events in the network. The packets can be dropped by network routers when queues are full, they can be delayed due to competing traffic, or they can arrive out of order. Packet loss and a lack of regularity in the packet stream at the receiving phone can result in poor sound quality.

Data that relate to the performance of IP telephony on the network in Figure 1 is used to illustrate various concepts throughout the article. (See also Bearden, Denby, Karacali, Meloche, and Stott 2002a,b and Karacali, Denby, and Meloche 2004 for other studies based on data collected from this network.) This figure shows parts of the Avaya corporate network, including 37 special communication endpoints that are labeled in the figure. (The network also includes thousands of regular endpoints [e.g., users] that connect and disconnect at different nodes that are not shown here.) The circles represent network routers, and the triangles represent the special communication endpoints. The three agglomerations correspond to the Asia Pacific region, the Europe/Middle East region, and the Americas. Note that one of the edges (close to Dubai) is shaded in black; we return to this when we discuss network monitoring techniques in Section 4.2.

Network engineers designing and running networks [either within corporate and campus environments or for internet service providers (ISPs)] need data collection and analysis tools to assess quality-of-service (QoS) measures, such as packet loss rates, delays, and jitter, and for doing network bandwidth calculations. This information is needed for several important reasons, including monitoring network performance and utilization over time, drilling into problems and finding their causes, detecting congestion, planning capacity and network provisioning, and ensuring compliance with service-level agreements. The needs are especially critical with real-time applications that require very high and sustained levels of quality, such as VoIP, video streaming, video conferencing, and online games.

TECHNOMETRICS, AUGUST 2007, VOL. 49, NO. 3
DOI 10.1198/004017007000000290

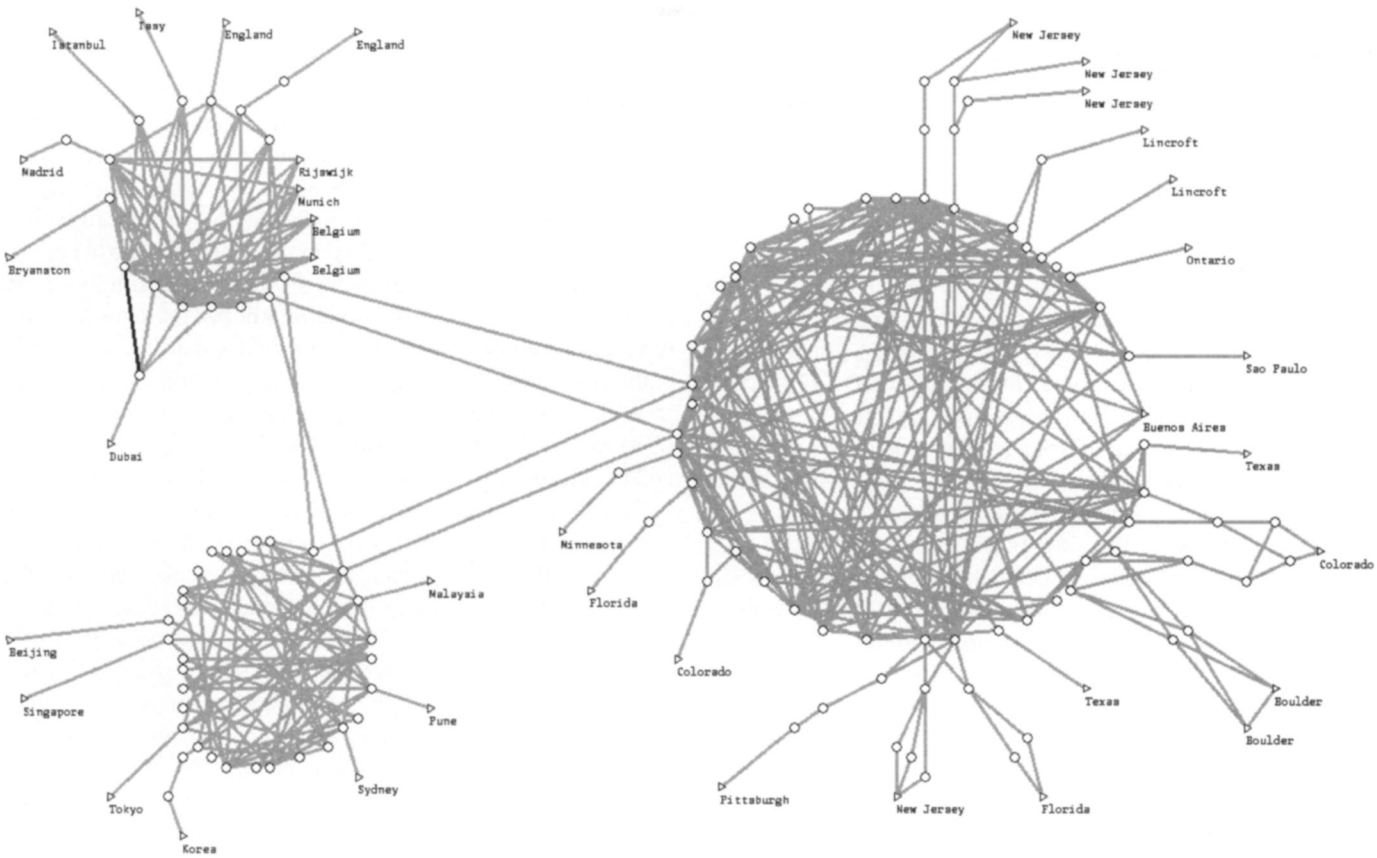

Figure 1. The Avaya corporate network.

This is a challenging problem for several reasons. First, the size of the network (often much larger than the one shown in Fig. 1) can limit the type of analyses that can be performed in practice. Second, networks are evolving entities, and QoS characteristics can change rapidly, for example, as a result of load or an automatic process that is attempting to circumvent some local network problem. Finally, network engineers often do not have access to all of the relevant components in the network, for example, a node that belongs to a different administrative domain or an edge that belongs to an ISP.

Traditional approaches to network analysis have relied on detailed queueing models at the individual router level. But such "local" modeling will not adequately capture the complexities and dynamic behavior of modern networks, including the fact that end-to-end results can be affected by interactions between nonadjacent network components. Expanding such local models to adequately incorporate the behavior of even a moderately sized network would be impractical.

There have been considerable efforts in recent years to identify appropriate sources of data and develop methods of data collection, analysis tools, and techniques to address these problems. This has led to the emergence of *network tomography*, a term first introduced by Vardi (1996) in the context of estimating the origin–destination (O–D) traffic matrix. Here we are interested in estimating the intensities of traffic flowing between the O–D pairs in the network. This information is important in network management, capacity planning, and provisioning. The challenge arises from the fact that only aggregate data on traffic counts at individual nodes (e.g., the number of packets going through a router) are available. The inverse problem is to recover distributions of traffic between all O–D pairs from the aggregate counts. Castro, Coates, Liang, Nowak, and Yu (2004a) has provided a review of this literature and an overview of developments in network tomography.

The focus in this article is on estimating and monitoring the QoS of the network by actively injecting test packets, or probes, into the network from nodes located on the periphery and collecting performance measurements on the injected probes. A primary goal is to estimate edge-level QoS characteristics from the end-to-end measurements. This inverse problem sometimes has been called *active network tomography*. Our overall purpose is to discuss a number of interesting statistical issues that arise in this area, including data collection, data quality, inference for the inverse problems, network monitoring, and the analysis of large-scale networks (graphs).

The article is organized as follows. Section 2 gives background on data networks, protocols, and network utilities that can be used for our purposes to provide measurements. Section 3 deals with estimation of the edge-level parameters from end-to-end path-level measurements, a very interesting inverse problem with many facets. It introduces a novel approach for edge-level estimation from end-to-end statistics based on a penalized likelihood that combines both traceroute and real-time transport protocol (RTP) data. Section 4 discusses several related topics, including methods for monitoring network performance over time and using the information to diagnose the edges or subnetworks where problems occur, and using visualizations to assess network performance. Section 5 briefly de-

scribes several issues involved in discovery, analysis, and visualization of network topology. Throughout the article, relevant literature and references are cited.

2. BACKGROUND AND DATA SOURCES

2.1 Network Preliminaries

A network can be represented by a directed graph $G = (V, E)$, where V is the set of nodes (routers) and E is the set of edges. Even though data networks can be specified at several different layers, this article concentrates on the IP layer or layer 3, in which a path consists of the source and destination nodes and the intervening IP routers. (For a detailed exposition on network layers and protocols, see Peterson and Davie 2003.)

Figure 2(a) is an example; it shows part of the University of North Carolina, Chapel Hill (UNC) network with 19 endpoints and 4 internal nodes. When data are sent from one location to another in a packet-switched network, the file's content is first broken into pieces, called packets. Additional information, such as origin–destination (O–D) information, reassembly instructions (such as sequence numbers), and error-correcting features, are added to the packet. The O–D information is used by the network routers and switches to deliver the packets to the intended recipient through the use of some network protocol. Figure 2(b) shows the paths from the Sitterson node (node 0) to all of the other endpoints on this UNC network. The paths are nothing more than a series of ordered edges that indicate the transmission route from a sender A to a receiver B.

Several different protocols control data transfer between communication endpoints, with each protocol fulfilling a particular need. For example, the transmission control protocol (TCP) guarantees reliable and in-order delivery of data from sender to receiver. On the other hand, the lightweight user datagram protocol (UDP) does not provide such guarantees. The internet control message protocol (ICMP) is used primarily to report error messages through the network. The ICMP defines several types of packets, including the echo request and echo reply messages commonly used by the ping utility and the "time-to-live exceeded" and "port unreachable" messages that are critical parts of the traceroute utility (see Sec. 2.2.1). The protocols can also rely on one another. For example, the well-known hypertext transfer protocol (HTTP) is built on top of the TCP protocol, which is built on top of the IP. The RTP for delivering audio (such as VoIP) and video over the internet is built on top of the UDP protocol because for these applications, it is not necessary to resend a packet that was dropped.

Many data communications and protocols involve roundtrip communications. This can be asymmetric; that is, the list of devices from point A to point B might not consist simply of the reversed list of devices from point B to point A. Even when network engineers intend to create a network that has symmetric paths (for various reasons that include simplicity), the routing protocols can create temporary asymmetries. Furthermore, even if the routing is symmetric, there are still two paths to consider, and the performance on these two paths can be quite different.

Packets arriving at a router or node are queued, awaiting their transmission to the next router according to the packet's protocol as handled by the router. Physically, a queue consists of a block of computer memory that temporarily stores the packets. If the queue (memory) is full when a packet arrives, then it is discarded; otherwise, it waits until it reaches the front of the queue and is forwarded to the next router on the way to its destination. This queueing mechanism is responsible for observed packet losses and, to a large extent, packet delays.

One of the challenges in analyzing network performance arises from the fact that the topologies can change over time. Network paths are determined by cooperating routers on the network, and the routers use more or less standard routing protocols. Network engineers also can manually impose static routes as desired, although such interventions are not common. The routing protocols work by exchanging messages that can result in changes to the paths on the network; these changes can occur within seconds or minutes, depending on the configurations of the routing protocols.

2.2 Data

2.2.1 Ping and Traceroute Data. Several existing data collection utilities can be used to collect data on the performance of network connections and remote computers. Here we discuss the ping and traceroute utilities and their usefulness.

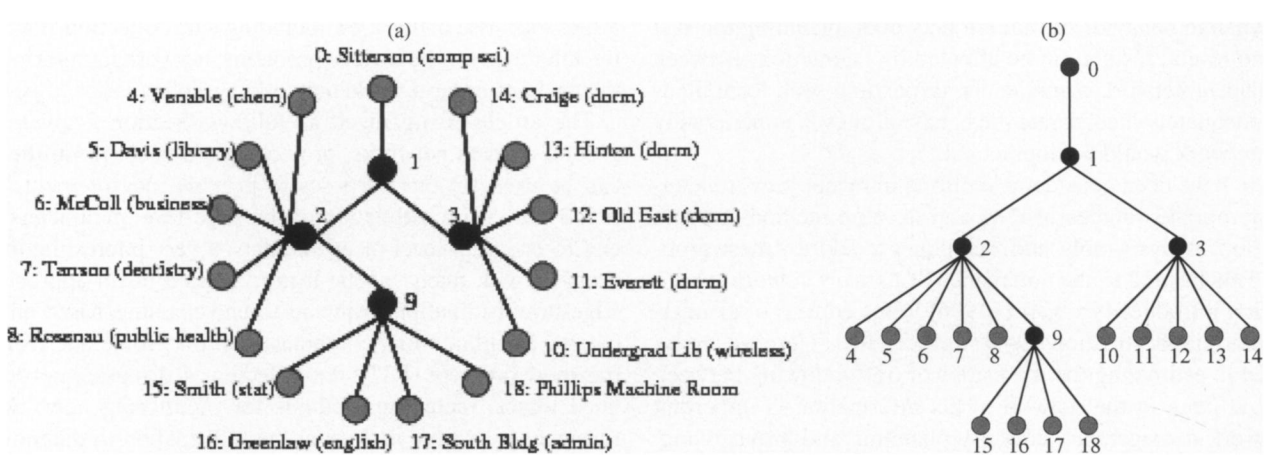

Figure 2. Schematic of the UNC network (a) and topology of the UNC network for the VoIP study (b).

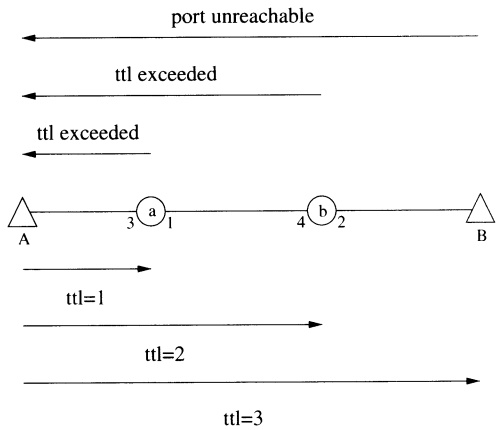

Figure 3. Traceroute session.

The ping utility is used by system administrators to check whether a remote computer is operating and to determine network connectivity. The source computer sends an ICMP packet to the remote computer's IP address. If the destination computer is up and the network connections are fine, then the source computer receives a return ICMP packet. Ping can be used to measure the amount of time it takes for a packet to make the complete trip. Thus data can be collected on roundtrip times and delays.

The traceroute utility is commonly used to identify network topology (see Sec. 5). Traceroute sends UDP packets from the source, and it exploits the time-to-live (TTL) field of a packet to determine the route that the packet takes to its destination (Fig. 3). Every IP packet has a TTL field that takes a value between 0 and 255. When a router receives an IP packet, it decrements this TTL field and forwards the packet to its destination according to its routing table. But if the TTL field was already 0, then the router sends an ICMP packet, indicating that TTL is exceeded, back to the source. Traceroute packets are sent at increasing values of TTL, starting with 1, until the destination is actually reached.

The source actually sends the traceroute packets to some invalid port at the destination. When the destination receives a packet destined for an invalid port, an ICMP packet indicating "port unreachable" is sent back to the source to indicate the error. The source then knows that the destination was reached. All of the previous packets failed to reach the destination because the TTL was too small, and the source received a "TTL exceeded" message from each of the intervening routers between the source and the destination, in the order in which they appeared. Figure 3 illustrates a traceroute session.

Many things can go wrong with traceroute and ping. Some routers are configured to not send or to not forward any ICMP messages. In addition, traceroute can produce false paths in the presence of per-packet load balancing, which sends each successive packet on a different path. Traceroute does not directly identify the routers, but identifies only the IP addresses. Routers have many IP addresses, one for each of their interfaces. When multiple paths are collected with traceroute, a given router may appear under different IP addresses in different paths. Thus the traceroute and ping data can provide incomplete or inaccurate information.

The biggest drawback with these data, however, is that their protocols are different from the applications of interest, and thus the network routers may treat these packets differently. Consequently, their performance is not necessarily a good surrogate for the applications under study, such as VoIP.

2.2.2 Injected Probe Data. A direct approach for application-sensitive monitoring is to actively inject "probe" packets that mimic the particular application (e.g., VoIP, FTP) into the network and measure various characteristics of end-to-end performance. The idea of injecting probe packets originated in the Multicast-based Inference of Network-internal Characteristics (MINC) project (Cáceres, Duffield, Horowitz, and Towsley 1999). (For more information on transmission mechanisms, see Peterson and Davie 2003.)

Figure 2(b) shows the topology that was used to collect data on the UNC campus network for VoIP readiness (Lawrence, Michailidis, and Nair 2006a). In this case the packets were sent from a single source node (Sitterson) to all the other endpoints on the periphery of the network. Other schemes are also common, such as sending packets from each node on the periphery to every other node (as in the Avaya network example discussed later).

The characteristics of interest include loss rates and delays of the probe packets. These can be one-way (node A to B) or roundtrip measurements (node A to B and back to A). The measurement of one-way delays is difficult because it involves synchronization of clocks at the sender and receiver. This is an important problem that we do not discuss here due to space limitations. Adhikari, Denby, Mallows, and Meloche (2003), Paxson (1998), Moon, Skelly, and Towsley (1999), Zhang, Liu, and Xia (2002), Jeske and Sampath (2003), and Jeske and Chakravartty (2006) have provided relevant algorithms.

There are many advantages to actively injecting probe packets to study network performance. First, probes can be sent proactively to monitor the network over time and detect developing problems before any adverse effect is experienced. Second, these probes measure QoS metrics on end-to-end performance for the applications of interest. These measures often depend on long-range interactions (in terms of network topology) on the network. An attempt at direct evaluation of all such interactions would be prohibitively expensive and quite unnecessary, because only a specific set of interactions is of relevance to any given application.

The key issue here is that the packets mimic the application of interest. In particular, the fields of the IP headers of the injected packets should be indistinguishable from those used by the actual application. In the presence of devices that perform deep-packet inspection, the IP payload also may need to be carefully crafted. To illustrate this, consider Figure 4, which shows roundtrip times for 100 UDP packets, and 100 ping (ICMP) packets sent during the same period to compare their performance. The solid circles represent ICMP packets, and the X's indicate UDP packets, both in milliseconds. All 100 UDP packets made the roundtrip, whereas only 73 of the ping packets completed it. The lost ICMP packets are indicated as vertical segments at the top of the figure. The UDP roundtrip times did not exceed 200 ms, whereas 27 of the ping roundtrip times were >400 ms. [In this case, there was a traffic shaper on the path between the source and the destination that occasionally

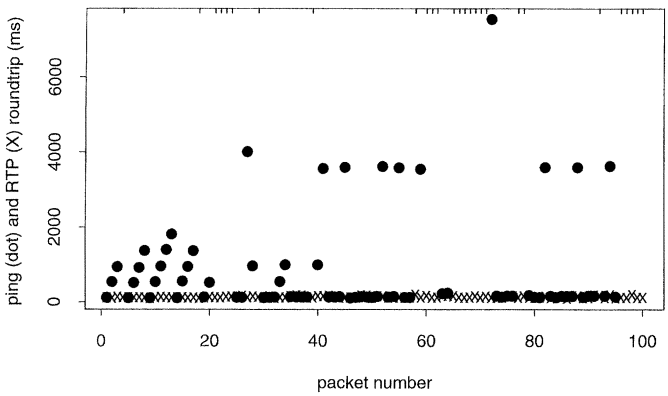

Figure 4. ICMP and UDP roundtrip times.

withheld some echo request packets (ICMP) for a multiple of .42 second delays.] The lesson to be drawn from these findings is that easily obtained measurements such as ping or traceroute are not necessarily adequate surrogates for the performance of other types of traffic.

Traceroute data are useful in identifying paths (i.e., connected edges over which the packets are transmitted from a source to a destination) and discovering network topology. The problem of topology discovery is discussed in Section 5. The use of traceroute data as auxiliary information for estimating edge-level QoS parameters is explored in Section 3.5.

In this article we restrict our attention to performance assessment based on injected packets. However, if we had access to network elements, such as routers, then we could address these problems by directly sampling packets. Estimation of network flow characteristics based on sampled data has been studied recently (see Claffy, Polyzos, and Braun 1993; Duffield 2004; Duffield, Lund, and Thorup 2005; Yang and Michailidis 2006).

2.2.3 Avaya Network: Data Collection and Summary. This section describes a study of the network in Figure 1 and summarizes the data. The network had 37 communication endpoints, for $N = 37 \times 36 = 1{,}332$ end-to-end pairs. For each pair, we sent an RTP stream of 100 packets from the source node to the destination. These packets were simply bounced back to the sender, leading to roundtrip delays and losses. The performance associated with these packets provided data on end-to-end network delay, jitter, and loss. Delay is caused primarily by the queues (congestion) at the routers. We can keep track of the delays for each packet, but in this study, we recorded only the median delay for the 100 packets. Jitter is a measure of variability that is given here by the interquartile range of the packet roundtrip times. Losses occur when the buffer is full and the packets are dropped. We computed loss as the percentage of packets (of the 100 packets sent) that did not make it back to the source within 5 seconds. (The worst network roundtrip time observed in the study was a bit under 1 second.) Finally, estimated mean opinion score, (eMOS; ITU P.800.1 2006) is a common qualitative measure of overall voice telephony performance. It is derived as a function of delay, loss, and jitter, and the values range from 1 to 5, with 5 being best.

The data for the 1,332 pairs were collected in 74 rounds of 18 pairs at a time in a random order over an 8-minute period. An endpoint was used in at most one pair in each round. To

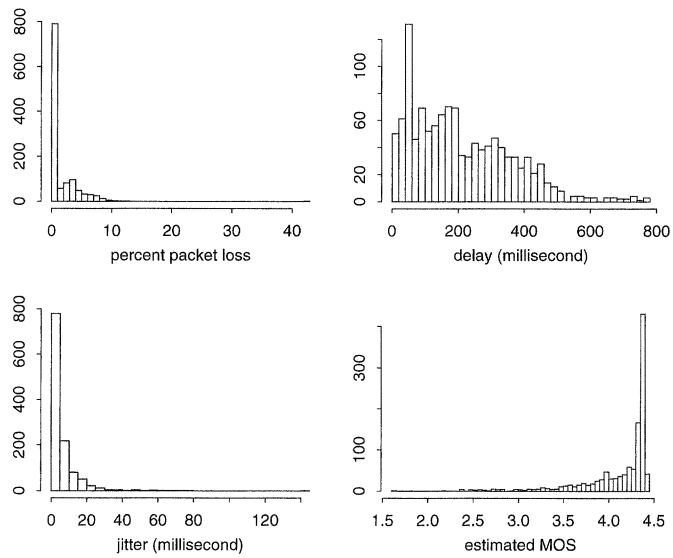

Figure 5. Summary of end-to-end performance data.

explore the temporal aspects of our problem, we collected data over a period of 2 weeks. Figure 5 shows the distribution of responses for the 1,332 pairs for one time slice and gives an overall picture of the quality of the network for this time period. The data for each end-to-end pair over time (for the entire 2 weeks) also can be analyzed to study temporal variation, time-of-day-effect, and so on. Such end-to-end data are very useful for monitoring network performance.

In addition to network performance, we are also interested in various features of the network itself, how they change over time, and other aspects. For example, Figure 6(a) shows the distribution of the path lengths (number of edges) for the 1,332 endpoint pairs for the first time slice. The lengths varied from 1 to 15 edges with average path length of 8. Figure 6(b) shows how the edges were distributed over the paths, that is, the number of source-destination paths to which each edge belongs. The median of this distribution is 12. About 20.6% of the edges belonged to a single path, whereas 3 edges belonged to more than 100 paths, indicating that the performance of these edges is critical for the network.

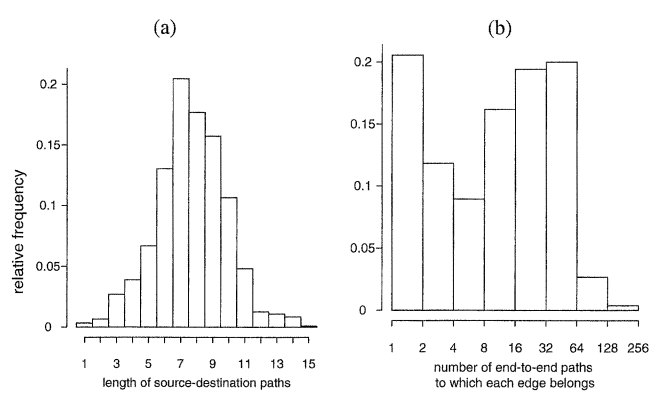

Figure 6. Path length distribution for the first routing matrix (a) and edge distribution for the first routing matrix (b).

3. ESTIMATING EDGE PARAMETERS FROM END–TO–END DATA: ACTIVE NETWORK TOMOGRAPHY

This section considers the problem of partitioning the end-to-end measurements to estimate loss rates and delay distributions of the edges. We use a combination of artificial and real situations and data collected on the Avaya network to illustrate the problems and discuss the modeling and analysis results.

3.1 Problem Formulation

For simplicity, we start with the delay estimation problem for the toy network in Figure 7(a). First, consider the case with a single source node 0 that sends probes to the two receiver nodes 2 and 3. Let X_1, X_2, and X_3 be the one-way delays associated with the edges as shown in the figure. For the moment, assume that the delays are fixed numbers. Let $Y_{\langle 0,2\rangle}$ and $Y_{\langle 0,3\rangle}$ denote the end-to-end delays associated with the paths $\langle 0, 2\rangle$ and $\langle 0, 3\rangle$. Then $Y_{\langle 0,2\rangle} = X_1 + X_2$ and $Y_{\langle 0,3\rangle} = X_1 + X_3$. Letting $\mathbf{y}$ and $\mathbf{x}$ be the corresponding vectors, we can write $\mathbf{y} = \mathbf{R}\mathbf{x}$, where

$$\mathbf{R} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}. \tag{1}$$

Here $\mathbf{R}$ is the routing matrix with $R_{ij} = 1$ if the ith path contains the jth edge and 0 otherwise.

Now the delays are not fixed and will vary from probe to probe. We assume a stochastic model where they are stationary and spatially independent; that is, the X_j's are independent. (We revisit these assumptions later in the section.) Suppose that we send M probes from 0 to each receiver node 2 and 3, for a total of $2M$ probes. Let μ_j be the mean delay and $\epsilon_{mj} = X_{mj} - \mu_j$, where X_{mj} is the delay at edge j experienced by the mth probe. Letting $\mathbf{y}$ be the $2M \times 1$ vector of end-to-end delays, we can write $\mathbf{y} = \mathbf{R}\boldsymbol{\mu} + \boldsymbol{\epsilon}$. Now the routing matrix $\mathbf{R}$ is $2M \times 3$, with each row in (1) repeated M times. In our analyses we have replaced the 100 individual packet delays by a single summary measure, the sample median. The goal then is to estimate the edge-level delays μ_1, μ_2, and μ_3 from the end-to-end delay data. Clearly, we cannot estimate all three edge parameters in this situation. We return to this estimability problem and related issues later.

We can use a different probing scheme that sends probes from each endpoint to all other endpoints, as shown in Figure 7(a). We now have six link-level parameters corresponding to the two directions of the edges. There are also six end-to-end delays, $Y_{\langle 0,2\rangle}$, $Y_{\langle 0,3\rangle}$, $Y_{\langle 2,0\rangle}$, $Y_{\langle 2,3\rangle}$, $Y_{\langle 3,0\rangle}$, and $Y_{\langle 3,2\rangle}$. The routing matrix $\mathbf{R}$ is now given by

$$\mathbf{R} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}. \tag{2}$$

Again, we have a linear model of the form $\mathbf{y} = \mathbf{R}\boldsymbol{\mu} + \boldsymbol{\epsilon}$.

The same formulation works in general for other networks. For the UNC VoIP study, the probes were sent from a single source node (Sitterson) to all of the other end-nodes using the topology in Figure 2(b). For the Avaya network shown in Figure 1, probes were sent from each of the 37 communication endpoints to all other 36 endpoints, for a total of 1,332 end-to-end pairs. There were 525 edges, so the dimension of the routing matrix was $1{,}332 \times 525$.

In general, we have a linear inverse problem of the form $\mathbf{y} = \mathbf{R}\boldsymbol{\mu} + \boldsymbol{\epsilon}$ where the goal is to estimate the mean link-level delays from the end-to-end data. (See Castro, Tsang, and Nowak 2004b and references therein for a review on linear inverse problems in network tomography.) If the routing matrix $\mathbf{R}$ is of full rank, then this is straightforward and can be solved using least squares, subject to the constraint that the μ's must be nonnegative. We also could use weighted least squares, which incorporates the variance–covariance structure of the error terms to get more efficient estimators.

In most cases, however, the routing matrix is not of full rank. For example, the rank of $\mathbf{R}$ in (1) is 2, and we have three edge parameters; in (2), the rank is 5, and we have six edges. For the network in Figure 1, the rank of $\mathbf{R}$ is 331, and there are 525 edges. Part of the degeneracy arises from a "chaining" phenomenon in which some edges are completely confounded with others (where an edge has only one child and the parent and child cannot be separated). If we remove such degeneracies, then $\mathbf{R}$ has 454 columns (edges), but its rank is still 331. Thus only a subspace spanned by the edge parameters is estimable. In the next few sections, we discuss alternative probing schemes and the use of auxiliary data to address the estimability problem.

Whereas estimation of mean delays is a linear inverse problem, inference for delay *distributions* is not. Specifically, let F_j be the distribution of X_j in the toy example in Figure 7(a). We

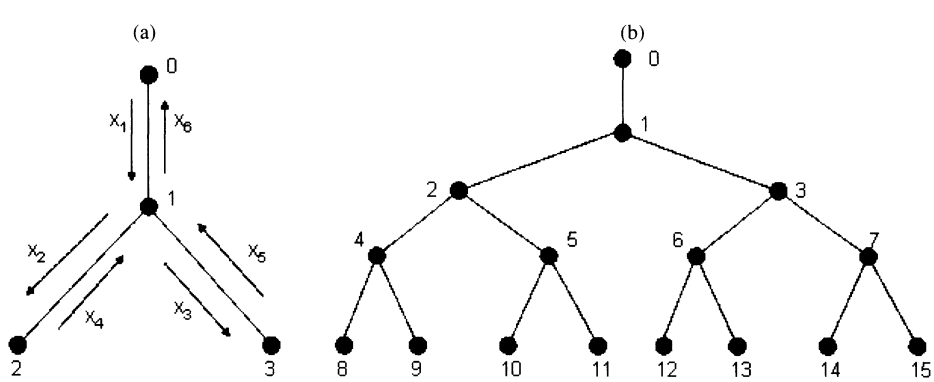

Figure 7. A two-layer binary tree (a) and a four-layer symmetric binary tree (b).

want to estimate $F_j, j = 1, \ldots, 6$, from the end-to-end data. This is a nonlinear deconvolution problem that is embedded within a graph.

The loss estimation problem can be transformed (approximately) into a linear problem. Consider again Figure 7(a). The X_j's are now binary with $X_j = 1$ if the packet is not lost on that edge and $X_j = 0$ if it is. The end-to-end data are $Y_{\langle 0,2\rangle} = X_1X_2$ and $Y_{\langle 0,3\rangle} = X_1X_3$, which are also binary. Let $\alpha_j = E(X_j)$, the probability of a successful transmission through edge j. Then $E(Y_{\langle 0,2\rangle}) = \alpha_1\alpha_2$ and $E(Y_{\langle 0,3\rangle}) = \alpha_1\alpha_3$. Suppose that we send M probes each from node 0 to nodes 2 and 3, and let $Z_{\langle 0,2\rangle}$ be the proportion of the M probes that successfully reach node 2; $Z_{\langle 0,3\rangle}$ is defined similarly. Let $Y = \log(Z)$ and $\beta_j = \log(\alpha_j)$. Then we can write an approximate linear model $\mathbf{y} = \mathbf{R}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, where $\mathbf{R}$ is the same routing matrix as before. (This is approximate, because the mean of the error terms is not identically 0 but tends to 0 as the number of probes $M \to \infty$.) Estimability issues for the loss rates are thus similar to those for estimating mean delays.

As mentioned in Section 2, there are other QoS measures of interest such as jitter and eMOS, but we do not consider them here.

The estimation methods discussed later assume, as is commonly done in the literature, that the losses and delays at the edges are temporally stationary and spatially independent. Temporal stationarity is reasonable because the probes are sent within an order of seconds, so the parameters will not vary much locally in time. By repeating the experiments over a specified period, we can estimate the QoS parameters as a function of time. A more troublesome assumption is that performance at different edges are independent of each other. This is unlikely to hold, because the subnetwork being studied can be part of a larger network with *cross traffic* that might influence behavior at several nodes simultaneously. There have been attempts to study this aspect through the use of network simulators (see Cáceres et al. 1999); however, the nature of the dependence is specific to the network being studied, and it appears difficult to obtain results that are generally applicable.

The rest of this section deals with several important and interesting statistical problems that arise in the context of estimating the loss rates and delays at the edge level. These include questions of estimability, design of probing experiments, inference methods for loss rates and delay distributions, and the use of traceroute data as auxiliary information for estimating edge parameters. We describe theoretical issues, previous research in the literature, and some new ideas applied to the probe experiments using the Avaya corporate network.

3.2 Estimability

The probing schemes that we have discussed so far send packets from a source node to a set of receiver nodes by transmitting the packets to one receiver at a time. This is called a *unicast* scheme. As we have seen, this leads to a routing matrix that is not of full rank, so we cannot estimate all the edge parameters. There is an alternative, called the *multicast* scheme, that gets around this problem by sending the packets simultaneously to a specified set of receiver nodes. Suppose that we want to send a packet from node 0 to receiver nodes 2 and 3 simultaneously for the network in Figure 7(a). The source node 0 sends a packet to node 1 where it is duplicated and forwarded to both nodes 2 and 3.

Consider again the loss problem, and suppose that we use the multicast scheme to transmit N packets from node 0 to nodes 2 and 3 simultaneously. The resulting sufficient statistics can be expressed as a 4-tuple, N_{11}, N_{10}, N_{01}, and N_{00}, which are the number of transmissions that reached both nodes 2 and 3, 2 but not 3, 3 but not 2, and no nodes. This 4-tuple has a multinomial distribution with parameters $(\pi_1, \pi_2, \pi_3, \pi_4)$ where $\pi_1 = \alpha_1\alpha_2\alpha_3$, $\pi_2 = \alpha_1\alpha_2(1 - \alpha_3)$, $\pi_3 = \alpha_1(1 - \alpha_2)\alpha_3$, and $\pi_4 = 1 - \alpha_1 + \alpha_1(1 - \alpha_2)(1 - \alpha_3)$. It can be seen that we now can estimate all three edge-level parameters from this scheme. The higher-order information from the multicast scheme (in the form of the shared experience of the top edge) is critical for estimability.

Even under this multicast transmission, mean delays or delay distributions are in general not estimable. To see this, consider again the toy example in Figure 7(a). Let the distribution of delay X_j be $\mathrm{N}(\mu_j, 1)$, for $j = 1, 2, 3$. Suppose, as before, that we send packets from node 0 to receiver nodes 2 and 3 simultaneously. Then the observed end-to-end delays are bivariate normal; the mean vector has elements $\mu_1 + \mu_2$ and $\mu_1 + \mu_3$, and the variance–covariance matrix has diagonal elements equal to 2 and off-diagonal elements equal to 1. Clearly, we cannot estimate all the μ_j's from these bivariate normal data. Similar problems also exist if we send packets from all end points to one another in this toy example or if we have larger networks.

Chen, Cao, and Bu (2007) showed that in general, all moments except the first of the delay distributions can be estimated using multicast data. More information or constraints are needed to estimate the first moment. One such case is when the higher moments are a function of the first moment, leading to additional estimating equations. To see this, suppose that the delay distributions $F_j(x)$ satisfy the property that $\mathrm{var}(X_j) = \mu^2(X_j)$. Then, for the two-layer tree in Figure 7(a), $\mathrm{cov}(Y_{\langle 0,2\rangle}, Y_{\langle 0,3\rangle}) = \mathrm{var}(X_1) = \mu^2(X_1)$, giving an estimating equation for μ_1, the mean delay of the first edge. Another situation in which we can estimate the delay distributions is when the edge delays have point mass at 0; that is, the delays can be 0 with positive probability. Consider again Figure 7(a) and the subset of end-to-end measurements $Y_{\langle 0,2\rangle} = 0$ and $Y_{\langle 0,3\rangle} = x$ for some positive value x. This implies that the delay at edges 1 and 2 are both 0, and thus $Y_{\langle 0,3\rangle} = X_3$. We can use these observations (for the various values of x) to estimate the delay distribution of edge 3. Similarly, we can use the subset of data with $Y_{\langle 0,2\rangle} = x$ and $Y_{\langle 0,3\rangle} = 0$ for various values of x to estimate distribution at edge 2. Thus all three edge-level distributions are estimable.

A major practical problem with using multicast schemes is that multicast support is not mandatory under IPv4, so many networks do not have multicast enabled by default (see de Goyan 1998). There have been proposals in the literature for using back-to-back unicast schemes, where packets are sent within nanoseconds of each other to two or more receivers to mimic multicast transmissions (Tsang, Coates, and Nowak 2003).

3.3 Design of Probing Experiments

Injecting probe packets into the network can add a significant amount of traffic, so we must carefully design probing experiments in terms of how much data to inject, when, for which pairs of endpoints, and so on. A full multicast scheme sending packets from many endpoints to many other endpoints in the network can generate more traffic than is desirable or necessary. We also may want to inject traffic in different parts of the network with different intensities over time.

We noted earlier that the higher-order information about the shared edges in multicast schemes is critical for estimating the internal edges. It turns out that second-order information (pairs of receiver nodes at a time or bicast schemes) is sufficient for this purpose. To understand this, consider Figure 7(b) and restrict attention to the three-layer tree with receiver nodes 4, 5, 6, and 7. A full multicast involves sending the packets from node 0 to all four receiver nodes simultaneously. This results in a $(2^4 - 1 = 15)$-dimensional multinomial experiment. Instead, it suffices to send packets to the pairs $\langle 4, 5\rangle$, $\langle 6, 7\rangle$, and $\langle 5, 6\rangle$, as we discuss later. Note that each of these bicast schemes results in only a four-dimensional multinomial experiment. In general, we can use a combination of unicast and bicast schemes to estimate all the edge-level parameters. This is particularly appealing if we are using back-to-back unicast schemes to mimic multicast schemes. Back-to-back schemes are most effective for pairs of receiver nodes and are less likely to mimic a multicast scheme as the number of receiver nodes increases.

A general class of flexible probing experiments (called flexicast) aimed at addressing the foregoing problem was introduced by Xi, Michailidis, and Nair (2006) and Lawrence et al. (2006a,b). It consists of a combination of k-cast schemes for different values of k, with each k-cast scheme aimed at studying a subnetwork. Each of the k-cast schemes by itself will not necessarily allow us to estimate the edge-level parameters of that subnetwork. The data must to be combined across the various k-cast schemes to allow estimating of the edge-level parameters. This class of experiments must satisfy some simple conditions for all of the edge-level parameters to be estimable. Xi et al. (2006) studied this problem for single-source tree topologies and showed that the following conditions are necessary and sufficient for identifiability of loss rates: (a) All receiver nodes are covered, and (b) for each internal node in the tree, there is a k-cast scheme that splits at that internal node. Lawrence et al. (2006a,b, 2007) showed that these conditions are also necessary and sufficient provided that the delay distribution is discrete or the higher-order moments are a function of the first moment.

As an example, again consider the three-layer binary tree. Suppose that we used an experiment with the two bicast schemes $\langle 4, 5\rangle$ and $\langle 6, 7\rangle$. We have covered all receiver nodes, and the first splits at node 2 and the second splits at node 3. However, there is no split at node 1, indicating that not all of the internal nodes are estimable. The following experiment with three bicasts, $\langle 4, 5\rangle$, $\langle 6, 7\rangle$, and $\langle 5, 6\rangle$, satisfy the identifiability conditions as the third pair splits at node 1. Note that we can replace the third pair $\langle 5, 6\rangle$ with another pair, such as $\langle 4, 7\rangle$, which also splits at node 1, indicating that the schemes satisfying the conditions are not unique.

A related question is how to allocate the total probe budget of, say, N probes among the various k-cast schemes. This can be formulated as an optimization problem using criteria in the optimal design literature. (See Xi et al. 2006 for discussion in the context of loss rates and single-source topologies.) It turns out that the optimal allocations depend on the unknown edge-level parameters.

A more interesting question relates to estimability with multisource topologies as in the Avaya network. For example, how should we supplement the unicast data (all end-to-end pairs) with a minimal number of bicast schemes (or back-to-back schemes) to estimate all of the internal edge parameters? In other words, given a routing matrix that is degenerate, can we characterize explicitly what probing experiments are needed to resolve the degeneracy? This is part of a more general question on studying the estimability problem with multisource topologies.

3.4 Inference Using Injected Probe Data

We assume that the probing schemes satisfy the identifiability conditions so that all of the edges are estimable. By treating the unobservable edge-level data as missing data, we can use the EM algorithm to compute the maximum likelihood estimators (MLEs) in both cases. This has been done extensively in network tomography applications (see Coates et al. 2002; Duffield, Horowitz, Lo Presti, and Towsley 2002). A different approach for estimating loss rates was introduced by Cáceres et al. (1999), where the sufficient statistics of the data are calculated and then a solution to the likelihood function is obtained by solving a set of polynomial equations. It can be shown that this approach leads to asymptotic MLEs, but their performance in finite samples can be inferior.

3.4.1 Delay Distributions. To keep things simple, we consider only situations with a single source, but the same ideas apply to mulitsource situations as well. Let X_k denote the (unobservable) delay on edge k, let p_r be the path from node 0 to receiver node r, and let $Y_r = \sum_{k \in p_r} X_k$, the cumulative delay accumulated along this path. We observe end-to-end delays consisting of Y_r for all of the receiver nodes.

Lawrence et al. (2007) examined inference for mean delays $\mu_j = E(X_j)$ under the model where $\text{var}(X_j) \propto \mu_j^{\phi}$ for some $\phi > 0$. ML estimation is still intractable even for simple parametric models, so they proposed and studied the properties of moment-based methods. For example, the covariance terms $\text{cov}(Y_r, Y_s)$ provide additional estimating equations for estimating the edge-level mean delays.

The more general case of estimating delay *distributions* has been studied in network applications assuming a discrete distribution. A simple and fast algorithm was developed by Lo Presti, Duffield, Horowitz, and Towsley (2002), but this is quite inefficient. Liang and Yu (2003) proposed a pseudolikelihood estimation method with multicast data. This involves using only the one- and two-dimensional data and ignoring the higher-order information for computational simplicity. Lawrence et al. (2006a,b) studied ML estimation and the behavior of the EM algorithm for general flexicast schemes. Again, the EM algorithm is a reasonable technique for computing the MLEs. However, the complexity of the EM algorithm (particularly when computing conditional expectations of the internal edge delays for each bin) is prohibitive for all but fairly small networks. To

deal with larger networks, Lawrence et al. (2006a,b) developed a grafting method that fits "local" EMs to the subtrees defined by the k-cast schemes and then combines the estimates through a fixed-point algorithm. This hybrid algorithm is fast and has reasonable statistical efficiency compared with full ML estimation. (See Chen et al. 2007 for delay estimation using Fourier methods and Tsang et al. 2003 and Shih and Hero 2003 for inference under other models.)

3.4.2 Loss Rates. The structure of the EM algorithm for loss estimation was studied by Xi et al. (2006). It works well for small to moderate-sized topologies but becomes computationally infeasible as the number of edges gets large. Here describe a new, scalable algorithm based on least squares that leads to fast estimation of loss rates (Michailidis, Nair, and Xi 2007).

For concreteness, consider the three-layer symmetric binary tree [Fig. 7(b) with just three layers and receiver nodes 4, 5, 6, and 7], and suppose that we use a full multicast experiment to all of the receivers (4, 5, 6, 7). There are 16 possible outcomes,

$$(1,1,1,1),(1,1,1,0),(1,1,0,0),\ldots,(0,0,0,0).$$

Denote the corresponding number of events for each outcome by $N_{(1,1,1,1)}, N_{(1,1,1,0)}$, and so on. We can ignore the last one, because there are only 15 linearly independent results. Consider the one-to-one transformation of these 15 events to the following: $(1,1,1,1),(1,1,1,+),(1,1,+,+),\ldots$, where $+$ indicates either a 1 or a 0. The new outcomes are obtained by replacing all of the 0's with $+$. Let $M_{(i,j,k,l)}$ denote the number of these outcomes. Now if N denotes the total number of probes for the experiment, then we can write $E(M_{i,j,k,l})$ as N times the product of appropriate link-level α's; for instance, $E(M_{(1,1,1,+)}) = N\gamma_{(1,1,1,+)}$, where $\gamma_{(1,1,1,+)} = \alpha_1\alpha_2\alpha_3\alpha_4\alpha_5\alpha_6$. Similarly, $E(M_{(1,+,+,1)}) = N\gamma_{(1,+,+,1)}$, where $\gamma_{(1,+,+,1)} = \alpha_1\alpha_2\alpha_3\alpha_4\alpha_7$. The expectations for the other M's can be written similarly as a product of a suitable subset of the α_k's. This naturally suggests fitting a log-linear model to the estimated probabilities $Y_{(i_1,\ldots,i_k)} = \log(M_{(i_1,\ldots,i_k)}/N)$.

Formally, suppose that we have single-source topology with P edges and M receiver nodes and that we send probes from the source to all of the receiver nodes using a multicast scheme. Then there are $2^M - 1$ outcomes. Let Y be the $2^M - 1$ column vector containing the logarithms of the estimated probabilities, let $\mathbf{R}$ be the $(2^M - 1) \times P$ routing binary matrix, let $\boldsymbol{\beta}$ be a P column vector of regression coefficients with $\beta_j = \log(\alpha_j)$, and let $\boldsymbol{\epsilon}$ be a column vector of "error" terms with $E(\boldsymbol{\epsilon}\boldsymbol{\epsilon}') = \mathbf{V}$. We then have the (approximate) linear model $\mathbf{y} = \mathbf{R}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, similar to the formulation in Section 3.1. Now we are also using the higher-dimensional outcomes, resulting in a full-rank routing matrix $\mathbf{R}$. [Castro et al. (2004a,b) also mentioned a linear model in terms of conditional probabilities, but the present formulation is more efficient.]

We can then estimate the parameters in the linear model using least squares methods. The ordinary least squares estimate $\boldsymbol{\beta}$ is given by

$$\hat{\boldsymbol{\beta}}_O = (\mathbf{R}'\mathbf{R})^{-1}\mathbf{R}'\mathbf{y}.$$

However, the error terms have unequal variances and are correlated, so it is more efficient to use generalized least squares. The form of $\mathbf{V}$, the variance–covariance matrix of $\mathbf{y}$, can be obtained in terms of the probabilities $\gamma_{(1,1,1,+)}$, $\gamma_{(1,+,+,1)}$, and so on, for example,

$$\mathrm{Var}\left(Y_{(1,1,1,+)}\right) = \frac{\gamma_{(1,1,1,+)}(1-\gamma_{(1,1,1,+)})}{N\gamma^2_{(1,1,1,+)}}$$

and

$$\mathrm{Cov}\left(Y_{(1,1,1,+)}, Y_{(1,+,+,1)}\right) = \frac{\gamma_{(1,1,1,1)} - \gamma_{(1,1,1,+)}\gamma_{(1,+,+,1)}}{N\gamma_{(1,1,1,+)}\gamma_{(1,+,+,1)}}.$$

A simple, noniterative generalized least squares estimator can be obtained as

$$\hat{\boldsymbol{\beta}}_G = (\mathbf{R}'\hat{\mathbf{V}}^{-1}\mathbf{R})^{-1}\mathbf{R}'\hat{\mathbf{V}}^{-1}\mathbf{y},$$

where $\hat{\mathbf{V}}$ is obtained using the method-of-moments estimates of γ. However, this simple plug-in estimate of $\mathbf{V}$ can perform poorly in small samples. A more efficient alternative is the iteratively reweighted least squares (IRWLS) estimator

$$\hat{\boldsymbol{\beta}}_I = (\mathbf{R}'\bar{\mathbf{V}}^{-1}\mathbf{R})^{-1}\mathbf{R}'\bar{\mathbf{V}}^{-1}\mathbf{y},$$

where $\bar{\mathbf{V}}$ is based on the estimated values of α from the past iteration. Recall that the γ's are products of appropriate subsets of the α's. We found the IRWLS estimators to be numerically very close to the ML estimators even in relatively small samples.

Least squares (LS) estimation with multicast experiments can be computationally expensive for large networks as the number of rows in the routing matrix $\mathbf{R}$ increases exponentially with the size of the network. The flexicast experiments discussed earlier are more attractive in this case. If we use a minimal number of bicasts that satisfy the identifiability condition discussed in the last section, then the number of rows of the entire routing matrix $\mathbf{R}$ in a flexicast experiment is linear in the size of the receiver set, as opposed to exponential for multicast experiments.

The LS estimation approach extends readily to flexicast schemes and multisource topologies. Specifically, for each k-cast scheme h, write the corresponding loglinear model $\mathbf{y}^h = \mathbf{R}^h\boldsymbol{\beta} + \boldsymbol{\epsilon}^h$. By stacking together the data and the routing matrices for all of the k-cast schemes in the flexicast experiment, we get a combined linear model $\mathbf{y} = \mathbf{R}\boldsymbol{\beta} + \boldsymbol{\epsilon}$. Because the different k-cast schemes are independent, the variance–covariance matrix of the error term given by

$$\mathbf{V} = \begin{bmatrix} V^1 & 0 & \cdots & \cdots & 0 \\ 0 & V^2 & \cdots & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \cdots & V^H \end{bmatrix}.$$

Another advantage of these LS-based schemes is that there is an explicit expression for the (asymptotic) variance–covariance matrix of the estimators, leading to easy construction of standard errors and hypothesis tests. The LS algorithms also lend themselves to easy updating as additional data become available.

We now describe some properties of these LS estimators (see Michailidis et al. 2007 for more details). The generalized least squares (GLS) and IRWLS estimates are consistent, asymptotically normal, and fully efficient, that is, $(\mathbf{R}'\mathbf{V}^{-1}\mathbf{R})^{-1} = \mathcal{I}^{-1}(\boldsymbol{\beta})$, where $\mathcal{I}^{-1}(\boldsymbol{\beta})$ denotes the inverse of the Fisher information matrix of $\boldsymbol{\beta}$. However, in small samples, the GLS estimators do not perform as well as the IRWLS estimators.

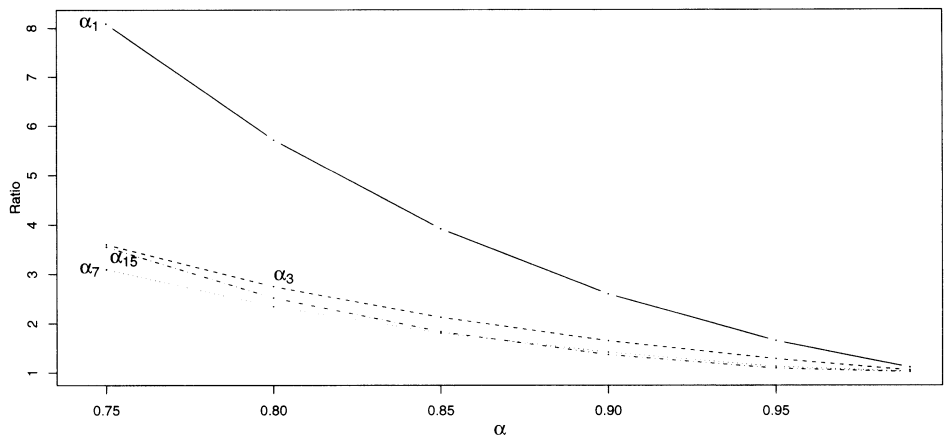

Figure 8. Relative efficiency of the LS estimators versus IRWLS.

Figure 8 shows the asymptotic relative efficiencies of the IRWLS estimators compared with the LS estimators for selected edges for a four-layer tree [see Fig. 7(b)]. The y-axis in Figure 8 are the relative efficiencies (ratio of asymptotic variances of the LS estimators and the IRWLS estimators) for four edges (1, 3, 7, and 15) as a function of the probability of successful transmission that varies from .75 to .99. The computations were done under a multicast scheme and assuming the probabilities for all of the edges are the same. We see that the LS estimators can be quite inefficient, and so the use of iterative schemes can have big payoff.

One issue that we have not discussed is the large number of parameters (loss or delay) to be estimated in a typical network and the fact that these parameters vary over time. Thus some type of regularization on shrinkage is warranted. This should take into account the network topology and other relevant information. Regularization has been used in network tomography, although in a different context. Zhang, Roughan, Lund, and Donoho (2003) used a *gravity model* in the O–D estimation problem to get around the estimability issues. (See Liang, Taft, and Yu (2006) for a variation of this approach.)

3.5 Combining RTP Probing With Traceroute Data

Because VoIP and other real-time applications are sent using RTP, an important need is to understand how RTP streams are handled by the network. For this purpose, we want the probes to be RTP streams or to be able to predict the behavior of RTP streams. Moreover, in corporate or large networks, the estimability problems are often severe, and multicast probing generally is not available. Thus we consider an alternative method for estimating the edge parameters based on combining unicast end-to-end RTP tests with additional data arising from traceroute tests. The ideas are also applicable to combining other types of testing data. In this section we restrict attention to mean delays and focus on roundtrip times; similar discussion holds for one-way delays and loss data.

The idea is that on the one hand, RTP data are available for end-to-end pairs but not directly for all edges. On the other hand, traceroute data are available for all edges but, for reasons discussed in Section 2.2.2, might not reliably mimic the behavior of RTP traffic at all times on all edges. We discuss various ways to combine the two sources of information to estimate the edge parameters relevant for VoIP.

To be concrete, consider again the simple two-layer network shown in Figure 7(a). Recall that the traceroute involves a sequence of probes with roundtrip times to successive routers along each path. So for the path $\langle 0, 2\rangle$, we get the traceroute data $Z_1 = X^{tr}_{1m} + X^{tr}_{6m}$ and $Z_2 = X^{tr}_{1n} + X^{tr}_{2n} + X^{tr}_{4n} + X^{tr}_{6n}$ for packets m and n. For path $\langle 0, 3\rangle$, we get $Z_3 = X^{tr}_{1p} + X^{tr}_{6p}$ and $Z_4 = X^{tr}_{1r} + X^{tr}_{3r} + X^{tr}_{5r} + X^{tr}_{6r}$ for packets p and r. A similar set of data is observed for the other end-to-end pairs $\langle 2, 0\rangle$, $\langle 2, 3\rangle$, $\langle 3, 0\rangle$, and $\langle 3, 2\rangle$. Letting $E(X^{tr}_j) = \mu^{tr}_j$, we can write a linear model for the traceroute delay data as before as $\mathbf{z} = \mathbf{R}^{tr}\mathbf{x} + \boldsymbol{\delta}$. The routing matrix is now given by

$$\mathbf{R}^{tr} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}. \tag{3}$$

This matrix has full rank ($=6$), so we can estimate all six mean delay parameters for traceroute performance, μ^{tr}_j's, directly. The parameters must be estimated under the nonnegativity constraint that $\mu^{tr}_j \geq 0$. The constrained least squares algorithm (NNLS) is an option, but it can be time-consuming with large networks. The pool-adjacent violators algorithm (Barlow, Bartholomew, Bremner, and Brunk 1972) is a faster alternative; we denote it as $\hat{\mu}^{tr}_{\text{pava}}$. This method pools adjacent points that violate the monotonicity constraint, averages them, and iterates until the sequence is monotonic. This provides an estimate for each edge in each traceroute path. If an edge is included in more than one path, then we take as the edge estimate the median of these values. In a sense, $\hat{\mu}^{tr}_{\text{pava}}$ amounts to taking care of the nonnegativity constraint on a path-by-path basis rather than taking care of it globally, as does NNLS. We have compared the two methods on data collected on the Avaya network (described in

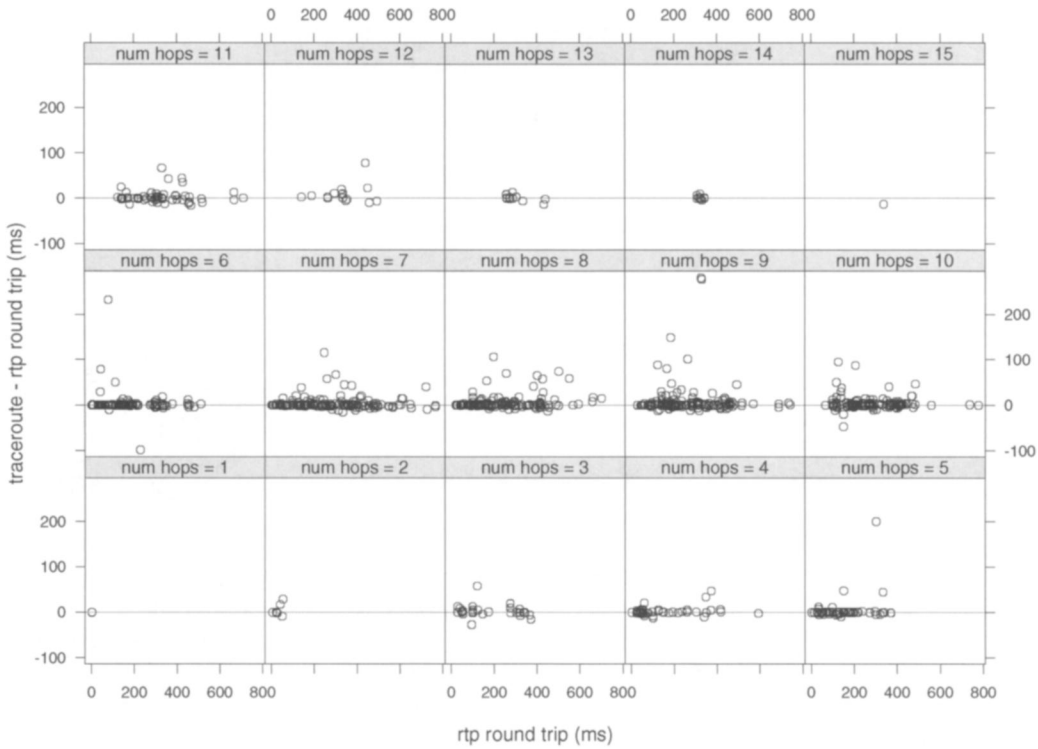

Figure 9. Comparing end-to-end traceroute and RTP.

Sec. 2.2.3) and found them to perform similarly for the most part. For computational reasons, we focus on $\hat{\mu}^{tr}_{\text{pava}}$ in the rest of this section.

Our goal is to estimate RTP delays. We investigate the possibility of using $\hat{\mu}^{tr}_{\text{pava}}$ as an estimate for μ^{rtp}. We also assess the relationship between the traceroute and RTP data to combine them. We have roundtrip delay times for all end-to-end pairs from both data sources. Figure 9 plots the differences for the 1,332 cases with end-to-end pairs grouped according to the number of edges between the source and destination. Most of the differences are near 0, with 63% within 2 ms, 25% between 2 and 10 ms, and 12% >10 ms. As expected from engineering considerations, traceroute is equal to or (in a small but not negligible set of cases) larger than the RTP measurement, but rarely smaller. The differences are substantial enough so that we cannot simply take the traceroute results as valid surrogates for RTP delays. We now discuss several heuristic methods that use the traceroute data as auxiliary information to estimate RTP performance.

A straightforward approach is to rescale the estimated traceroute delays $\hat{\mu}^{tr}_{\text{pava}}$ so that the estimated delay summed over all of the edges in the end-to-end path matches the RTP delay measured on the same end-to-end path. When an edge is a member of more than one end-to-end path, we first scale on each path and then take the median over all of the paths that contain this edge. We call this estimator $\hat{\mu}^{rtp}_{\text{cal}}$, where "cal" designates "calibration."

An alternative is a penalization framework that uses the intermediate values $\hat{\mu}^{tr}_{\text{pava}}$ to estimate the μ^{rtp}. This can be obtained by minimizing

$$\|\mathbf{y} - \mathbf{R}\boldsymbol{\mu}^{rtp}\|^2 + \lambda\|\boldsymbol{\mu}^{rtp} - \hat{\boldsymbol{\mu}}^{tr}_{\text{pava}}\|^2, \tag{4}$$

where λ is a weighting parameter that is specified. The resulting estimator, $\hat{\boldsymbol{\mu}}^{rtp}_{\text{pen}}$, can be obtained using LS. We note in passing that regularization methods have been used in network tomography in other contexts as well.

We also can directly minimize

$$\|\mathbf{y} - \mathbf{R}^{rtp}\boldsymbol{\mu}^{rtp}\|^2 + \lambda\|\mathbf{z} - \mathbf{R}^{tr}\boldsymbol{\mu}^{rtp}\|^2, \tag{5}$$

which is just a weighted LS, with λ representing $\text{var}(Y)/\text{var}(Z)$. Because traceroute delays can be substantially larger than RTP delays, it makes sense to take λ small or even to consider the limit as $\lambda \to 0$. In this limiting case, we are estimating all estimable linear functions from the RTP data alone while using the traceroute data solely to disambiguate the degeneracies. This approach goes some way toward allowing for the possibility that the traceroute observations are not estimating the same parameters as the RTP data.

Figure 10 compares the three edge-level estimators $\hat{\mu}^{tr}_{\text{pava}}$, $\hat{\mu}^{rtp}_{\text{cal}}$, and $\hat{\mu}^{rtp}_{\text{pen}}$ from (4) with $\lambda = .001$. Other small values of λ would produce nearly identical fits. The scatter diagrams show the values of $\hat{\mu}^{tr}_{\text{pava}} - \hat{\mu}^{rtp}_{\text{cal}}$ against $\hat{\mu}^{rtp}_{\text{cal}}$ and $\hat{\mu}^{rtp}_{\text{pen}} - \hat{\mu}^{rtp}_{\text{cal}}$ against $\hat{\mu}^{rtp}_{\text{cal}}$ for each of the 525 edges. The first two estimates, in panel (a), are in fairly close agreement; both have 114 coefficients (22%) that are exactly 0, they differ by at most 40 ms, and 99% of them are within 10 ms of one another. These estimators are quite different from the penalized estimator $\hat{\mu}^{rtp}_{\text{pen}}$, as seen in panel (b).

Figure 11 provides a different comparison of the same three estimators. The scatter diagrams display residuals against fitted values, and the mean sum of squared residuals are 207, 217, and 76. Using the second penalized approach in (5) also leads

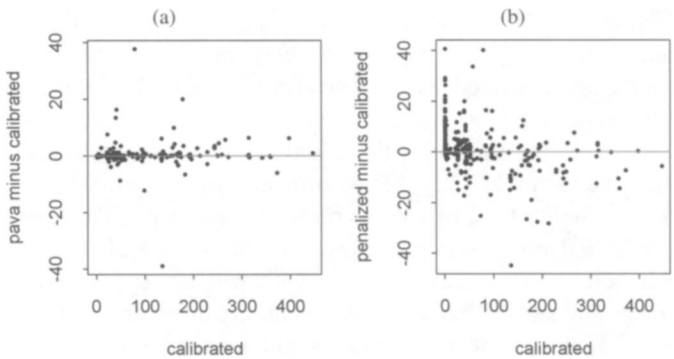

Figure 10. Comparing $\hat{\mu}^{rtp}_{\text{cal}}$ to $\hat{\mu}^{tr}_{\text{pava}}$ (a) and $\hat{\mu}^{rtp}_{\text{pen}}$ (b).

to a mean sum of squared residuals of 76. The penalized estimators are clearly superior, and using $\hat{\mu}^{rtp}_{\text{pen}}$ from (4) involves much less computation than using (5).

An important practical problem not addressed here is computing the estimators in real time in a distributed network environment. The penalized estimation method seems to provide a good balance of accuracy and computational ease, but further research is needed.

A more general formulation for combining the two sources is a calibration model. Recall that in the calibration problem, we have inexpensive but less reliable measurements from one source and expensive, reliable, but limited measurements from another. In our case, the traceroute is the less reliable source and the RTP test is the more reliable source. We have end-to-end delay measurements for the 1,332 pairs from both sources but only traceroute data for the edges. Thus we can develop a calibration model that relates the two data sources from the end-to-end measurements and use the model to calibrate the edge-level traceroute data. Using the existing notation, we can write

$$Y_i^{tr} = f(Y_i^{rtp}, \mathbf{R}|p_i)$$

for $i = 1, \ldots, n$. The functional form of $f(\cdot)$ captures the relationship between the end-to-end RTP data Y_i^{rtp} and the end-to-end traceroute data. This relationship may depend on the path p_i, for instance, through the number of edges in the path. There are many ways to specify and fit $f(\cdot)$. Once this is done and the model is fit, the information from traceroute and RTP data can be combined to estimate the edge parameters.

4. MONITORING NETWORK PERFORMANCE: DETECTING AND DIAGNOSING CHANGES

This section considers methods for monitoring network performance, detecting degradation in performance levels, and diagnosing where the problems occur. Given the high-level of quality, network monitoring often amounts to filtering through large amounts of irrelevant data to determine unusual behavior that is important.

4.1 Monitoring Techniques

Because our primary interest is in end-to-end performance, for the purposes of monitoring, we can focus our attention on path characteristics. There is an extensive literature on process monitoring techniques and changepoint detection methods that can be used to monitor end-to-end performance (see, e.g., Basseville and Nikiforov 1993; Stoumbos, Reynolds, Ryan, and Woodall 2000). Some preliminary results for monitoring loss rates using exponentially weighted moving average (EWMA) techniques have been given by Xi et al. (2006).

A challenge in implementing these techniques to network monitoring is that there is usually a large number of paths to monitor (e.g., 1,332 end-to-end pairs in the moderate-sized Avaya network). Monitoring a large number will result in a high rate of false alarms, but if we control the overall false alarm rate, then the power of detection will be low. It is important to keep the number of paths to be monitored to a reasonable number (say 20–30) using engineering knowledge about the network topology and other critical business information. There are also key questions about the parameters to monitor, such as loss rates, mean delays, probability of large delays, or some overall measure, such as eMOS.

In cases where we can estimate the edge-level parameters (e.g., using one of the techniques discussed in Sec. 3), we can

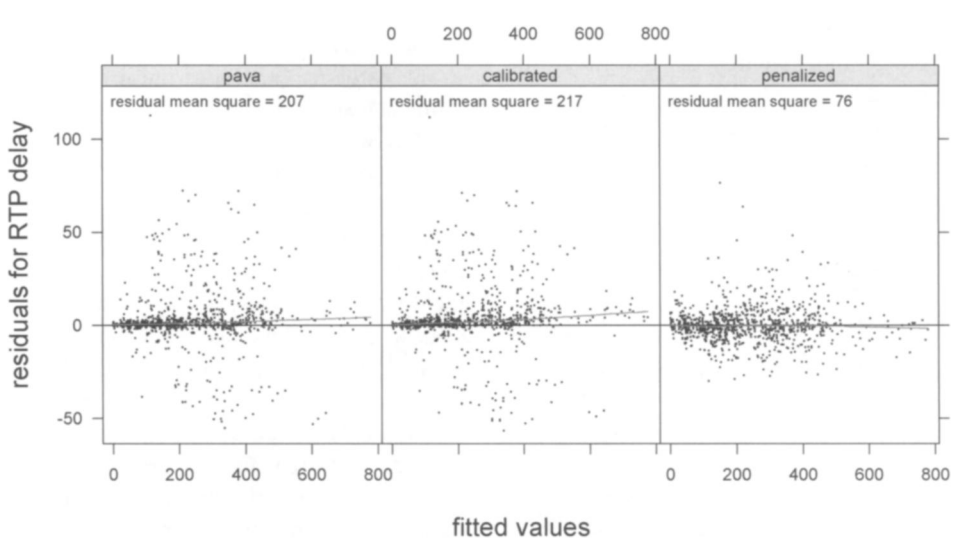

Figure 11. Comparing $\hat{\mu}^{rtp}_{\text{cal}}$ to $\hat{\mu}^{tr}_{\text{pava}}$ and $\hat{\mu}^{rtp}_{\text{pen}}$.

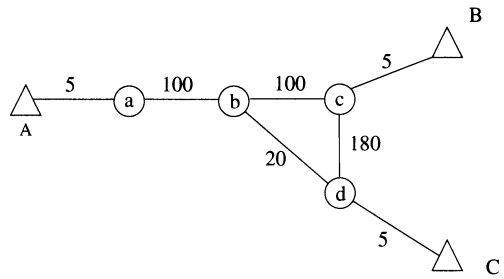

Figure 12. Root cause analysis.

also monitor these parameters directly. A statistical comparison of the relative efficiencies of monitoring techniques based on end-to-end paths versus edge parameters has been done by Yang (2006). However, this involves solving the inverse problem at each stage, which can be computationally expensive.

While detecting and diagnosing changes in performance, it is important to distinguish between changes that may be statistically significant and changes that are practically important for the network and its applications. For example, there can be a substantial change in the mean delay of one edge, but if this edge is part of a short path or if the other edges in the path have very small delays, then the overall path-level performance can still be quite adequate.

Consider Figure 12, which shows a simple topology with three endpoints (A, B, and C) involved in end-to-end communications that are handled by four routers (a, b, c, and d). Suppose that edge-level estimation resulted in the estimates (in milliseconds) next to each edge in the figure. Assume that the threshold beyond which end-to-end communication is inadequate is 200 ms. The edge c–d has the largest latency (180 ms) and thus might stand out from an edge-level estimation or monitoring perspective. But it is used only in the B–C communication, for which it is adequate. Thus the c–d edge does not constitute a problem from this perspective. On the other hand, the A–B communication is inadequate, because it has an accumulated latency of 210 ms. It is the serial utilization of the a–b–c edges that generates the problem. For example, users might experience a greater impact if the delay on a–b increased from 80 to 100 ms than if c–d increased from 120 to 180 ms.

Consequently, monitoring and diagnosing a network's performance must incorporate fundamentally the design and desired behavior of the network and use these factors as a basis for any alarms, alerts, or diagnostic statements. For the example of Figure 12, it could be argued that the network as designated is not capable of meeting the requirement for this application. To understand the possible root causes for end-to-end problems, we also must incorporate the paths into the analysis directly.

4.2 Diagnosing Changes: Root-Cause Analysis

With the foregoing example in mind, we propose the following approach for root-cause analysis. Let Y_i be an indicator of success and failure for the ith test, where $Y_i = 1$ means that the test fails and $Y_i = 0$ corresponds to the test being successful. This indicator could represent, for example, whether or not the delay for the ith test is adequate relative to its own threshold. As in Section 3.1, let $\mathbf{R}$ be the relevant routing matrix where $R_{ij} = 1$ if the jth edge is involved in the ith test and 0 otherwise for $j = 1, \ldots, K$ and K is the total number of edges. (This approach also can be generalized to incorporate not just topological factors such as edges but also other factors, such as software version or codec, etc.)

Let π_i be the probability that the ith test fails. For the jth edge, let p_j be the probability that performance on this edge "fails"; that is, the jth edge performs poorly enough so that the whole end-to-end test fails. Furthermore, a test has some probability of failing that is not necessarily associated with any specific edge, so let p denote this background probability of failure; that is, p is the probability that the test fails, although none of the edges directly "causes" the failure. Then $Y_i \sim \text{Bernouilli}(\pi_i)$. For the end-to-end test to be successful, it must be successful for each traversed edge as well as for the background, so that

$$(1 - \pi_i) = (1 - p) \times \prod_{\{j: R_{ij}=1\}} (1 - p_j).$$

Estimating the p_j's is difficult mostly because of the large number of boundary conditions, but a straightforward approach appears to work well when there are few failures. We define the edge culpation ratio

$$I_j = \frac{\text{cardinality}\{i: y_i = 1 \text{ and } R_{ij} = 1\}}{\text{cardinality}\{i: y_i = 0 \text{ and } R_{ij} = 1\}};$$

that is, I_j is the odds ratio of the number of tests crossing edge j that failed to the number of tests crossing this edge that succeeded. We call this approach "culpation." The inculpation set is the set of edges for which the culpation ratio I_j is above some specified threshold. Such edges are inculpated as those that seem to contribute to a high probability of failure for tests traversing them. The inculpation set is simple to obtain and requires processing that can be easily distributed across the network. Indeed, the edge memberships in the routing matrix $\mathbf{R}$ can be determined and stored independently of the test results; distributing the culpation estimation process amounts to partitioning $\mathbf{R}$ into subsets of columns (i.e., edges) that are handled separately. Moreover, the culpation approach also can be applied to interactions among edges by expanding the routing matrix $\mathbf{R}$ and including products between the columns, as with analysis of variance. However, one needs to be judicious and include only interactions among edges that make some sense in the network; otherwise the problem could quickly become unmanageable.

We now illustrate how the inculpation method works using the eMOS statistic (see Sec. 2.2.3) as the end-to-end test, incorporating delay, loss, and jitter. We take the threshold for test failure to be 3.5 or lower, where 4.0 or higher is interpreted roughly as "toll quality" voice transmission. Figure 1 shows the inculpation set corresponding to the edges $\{j: I_j > .8\}$ on a gray scale as dark and the excluded edges as light. There is only one dark edge, involved in some of the test traffic in and out of the Dubai site but not all destinations from Dubai. We selected the threshold .8 as a value large enough so that the set contains only one edge. Decreasing the threshold results in including more edges. By varying the threshold and examining how the inculpation set changes, we can get an idea of the edges that seem to have the most problematic behavior. In this case, lowering the threshold adds further edges near Dubai.

4.3 Visualization Tools for Monitoring and Diagnosis

The formal techniques discussed so far are insufficient for understanding and diagnosing problems in large, complex networks. We have developed and used visualization tools in an iterative manner to supplement the formal techniques (Adhikari et al. 2006; Adhikari, Denby, Landwehr, and Meloche 2007). A static medium such as a journal article obviously constrains our ability to illustrate the features of a visualization tool.

The first challenge is to represent the topology and layout of a large network as a graph. We discuss this problem in more detail in the next section. Given such a graph, we have found animation to be useful in studying changes in performance over time. For the topology in Figure 1, we can color edges that exceed performance thresholds in, say, red and thus indicate the changes over time. Such animations will reveal coincidences over time and space that otherwise would be difficult to detect. One difficulty with this approach is that the network topology can change over time as nodes and edges are added and deleted, which in turn affects connectivity patterns and other characteristics that might be shown on the network graph.

One useful feature is to display additional information through mouse-over features. This could include static information that is too extensive to display all of the time in the plot. Because the topology graph can get very cluttered, we have found it better to reserve a separate panel in the plotting region to display such information than to display it directly adjacent to the item that is moused-over. For nodes, information could be such items as IP addresses for a router, its name, product type, and so on. For edges, the additional information could be a statistic measuring recent performance on that edge, such as delay.

An analysis of edges can hide information about end-to-end paths by visually emphasizing edges rather than paths. Thus it is important to reintegrate end-to-end path information into the visualization. A feature that we have developed for dealing with this is to click on two endpoints, after which the display highlights (e.g., colors differently) the path (or paths) taken by traffic between these two endpoints. The associated numerical results and graph coloring also can be restricted to calculations using only data from the end-to-end paths between the two endpoints.

Another feature is a pop-up time series plot that helps with all three aspects of drill-down, temporal behavior, and paths. Clicking on an edge creates a new plot, such as the one given in Figure 13, which shows end-to-end test results for delay for tests that traversed this edge over a period of time, regardless of the end-to-end paths. In Figure 13, time is shown on the x-axis with the most recent points on the right, so animating this over time gives the appearance of the points scrolling to the left. The label for the y-axis is shown on the right, which is unconventional but useful here because we typically are most interested in reading off numerical values for the most recent points. Furthermore, there are check boxes on the bottom that indicate all of the end-to-end paths that pertain to these data; by checking each box or not, we include or exclude in the plot the data points from that path. Sometimes a "banding" feature is visible in the plot, whereby the points fall in several separated horizontal bands. By checking and unchecking the boxes, interesting behavior can often be associated with specific end-to-end paths. Color also can be used to identify characteristics of the points, such as whether or not they contributed to triggering an alarm for the edge.

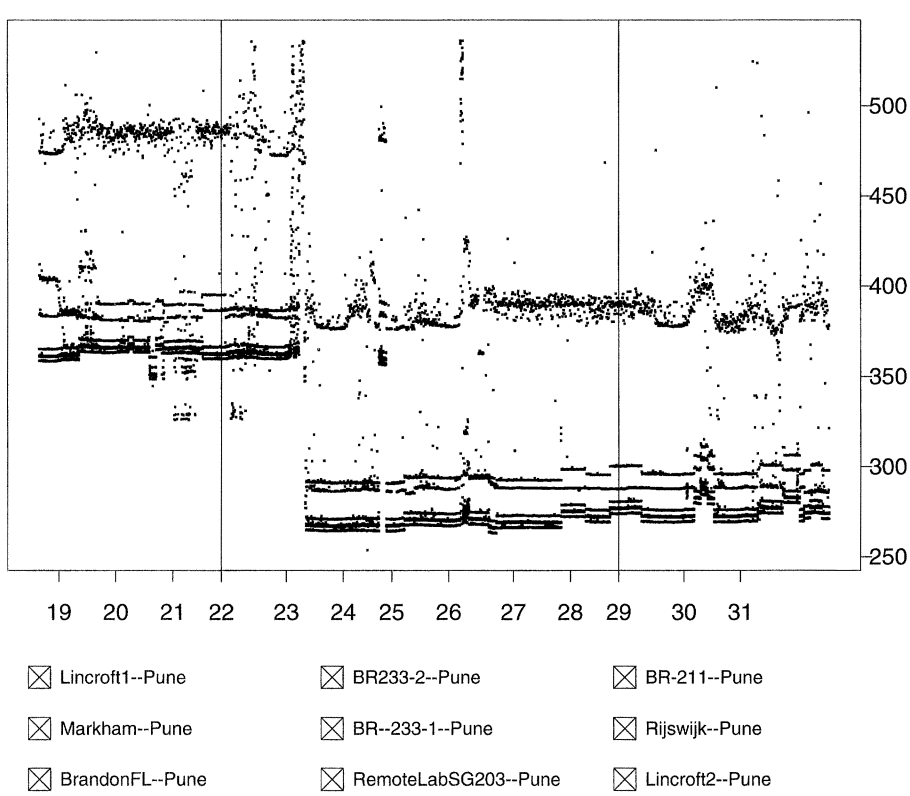

Figure 13. Pop-up time series for all paths touching the edge.

Clearly, there are many other possibilities for visualizing network performance and network data. Developing and implementing visualization tools that are useful to the network engineer or analyst requires careful consideration of many aspects, including user preferences and interactions. Some innovative visual displays for packet header data have been given by Solka, Marchette, and Wallet (2000).

5. ANALYSIS OF NETWORK TOPOLOGY

This section briefly describes several research issues associated with the analysis of network topology.

5.1 Topology Discovery

As has been evident throughout the article, knowledge of network topology is essential for assessing and monitoring its performance. However, this information is often difficult to obtain. Furthermore, the topology can change frequently as network devices go up or come down.

There are standard engineering methods for obtaining topological information provided that we have the necessary access. The most straightforward is to obtain the routing table of all the routers of interest. This information will give the connectivity graph of the network under consideration and can be obtained through using a certain protocol, SNMP. But this is a laborious task because of policies that restrict such router queries for security reasons. In addition, SNMP access usually receives low priority, and thus discovery based on this mechanism can take a long time (e.g., hours) for a large network. Furthermore, if routing protocols are active in the course of the discovery process, this can produce an inconsistent set of routing tables from which it will not be possible to recover some of the network paths.

Another approach is to use the IP record route mechanism, which involves setting a single bit in the IP header. Routers that process packets with the IP record route bit set will record their IP address in the IP header before sending the packet on its way to the destination. The IP header can record up to nine IP addresses, including the source IP address. However, several issues make IP record route difficult to exploit in practice. First, IP record route fails when there are more than eight intervening routers, which is common. Second, many devices routinely drop packets that have such special processing bits set. Third, and most importantly, many devices handle the IP record route packets in an entirely different way than regular packets, often sending such packets to a different interface than regular packets would have been sent. Thus the recorded route may not be the one actually used.

The approach that we used to discover the Avaya network topology in Figure 1 was based on traceroute data. We used traceroute data sent from all the end-to-end pairs and reconstructed the topology by merging the underlying spanning trees. Given the problems noted with traceroute data, this is a "best-guess" effort at constructing the topology. (See also Achlioptas, Clauset, Kempe, and Moore 2005 and DallAsta et al. 2006 for biases in using traceroute data.)

Methods also have been proposed in the literature for discovering a single-source topology using multicast (or back-to-back unicast) probe data. Given a source and a set of destinations, the problem is to identify a tree topology that best fits the data from among all topologies. We can define a similarity measure based on loss rates or delay times (nodes that share a longer common path should be more correlated than those with small paths), then use a clustering algorithm to group the nodes together and determine a topology. Duffield et al. (2002) showed that this strategy can completely identify a binary tree topology. Coates et al. (2002) discussed an alternative approach based on a random search strategy for locating an optimal tree topology. They also proposed a different probing method (called *sandwich probing*) that induces higher correlation and also gets around the clock-synchronization problem. Duffield and Lo Presti (2004) presented an alternative methodology based on measurements from end-to-end delay covariances. Finally, Castro et al. (2004a,b) discussed how the topology discovery problem can be cast as a ML estimation problem and provided some additional references.

The topology discovery problem as formulated herein is computationally rather difficult, and there is still no satisfactory solution. It has a similar flavor to identifying phylogenies in genetics, for which various clustering algorithms can be useful, although the phylogenic problem has more structure due to evolutionary considerations.

5.2 Representing, Visualizing, and Summarizing Networks

Representing a large, complex network as a graph is a challenging problem. The graph layout needs to combine some aspects of logical and physical reality. Achieving this by manually moving points around can be very time-consuming and tedious. Automatic layout procedures have received much research attention in the last few years (see Michaildis 2006 for a review).

The automatic layout problem is defined as follows: Given a set of nodes connected by a set of edges, identify the positions of the nodes in some space and calculate the curves that connect them. Most graph-drawing techniques use straight lines to connect the nodes and Euclidean space, although other choices, such as lattices or hyperbolic space, have proven useful in some application areas. Most scalable graph drawing algorithms use either an embedding model or an adjacency model (Michailidis 2006). In the former approach, path length distances are defined between the nodes, which are subsequently approximated by Euclidean distances derived from low (usually two- or three-)dimensional configurations. Multidimensional scaling and its variations (Buja and Swayne 2002) are examples of this approach. In the adjacency model, the emphasis is on placing close together in Euclidean space nodes that are connected together. A popular algorithm uses an eigenvalue decomposition of the Laplacian matrix of the underlying graph.

Whereas automatic layout algorithms are needed to get started, our experience is that there is generally detailed engineering, geographic, or network information that can help modify the layout and make it more interpretable for the user. We have found it beneficial to combine some automatic layout to get started with capabilities to manually modify the results.

A related problem is that the network can be so large or dense such that it all cannot, be displayed conveniently and informatively on the computer screen. Thinking of the virtual display as a large canvas features are needed to enable the user to navigate as desired across different regions of the canvas and to zoom in or out. A companion topic involves compression techniques to automatically reduce the size of the graph while at the same time enhancing semantically relevant information, such as the presence of highly connected nodes (hubs) and clusters, or preserving the shape of the node degree distribution. Several compression schemes that use node-degree or shortest-path importance, or node similarity measures, have been discussed by Adler and Mitzenmacher (2001) and Gilbert and Levchenko (2004).

As we have noted, changes in network topology over time add to the difficulties in representing and visualizing the network. Technical challenges include finding an appropriate data representation for dynamically evolving graphs, tracking those changes over time, updating its structure, and visualizing its evolution. Overviews of some relevant issues have been provided by Cortes, Pregibon, and Volinsky (2003) and Eppstein, Galil, and Italiano (1999).

6. CONCLUDING REMARKS

We have provided a review of several interesting statistical issues that arise in the context of assessing and monitoring network performance as well as in characterizing the properties of networks. Although considerable work has been done in this area, mostly in the network community, many interesting and challenging statistical problems remain. Because the research issues change rapidly with advances in technology, it is important that statisticians identify and collaborate with network engineers who are closely tied to the technology and problems.

ACKNOWLEDGMENTS

The authors thank two anonymous referees for useful suggestions. The research of Michailidis and Nair was supported in part by National Science Foundation DMS grant 0500535.

[Received November 2006. Revised March 2007.]

REFERENCES

Achlioptas, D., Clauset, A., Kempe, D., and Moore, C. (2005), "On the Bias of Traceroute Sampling, or Why Almost Every Network Looks Like It Has a Power Law," in *Proceedings of the 2005 Symposium on the Theory of Computation (STOC)*, Baltimore, MD, pp. 694–703.

Adhikari, A., Bianco, S. V., Denby, L., Mallows, C. L., Meloche, J., Rao, B., Sullivan, S. M., and Vardi, Y. (2006), "Distributed Monitoring and Analysis System for Network Traffic," U.S. Patent 7,031,264.

Adhikari, A., Denby, L., Landwehr, J. M., and Meloche, J. (2007), "Using Data Network Metrics, Graphics, and Topology to Explore Network Characteristics," in *Statistical Inverse Problems*, eds. R. Liu, W. Strawderman, and C. H. Zhang, Hayward, CA: IMS, to appear.

Adhikari, A., Denby, L., Mallows, C., and Meloche, J. (2003), "Measuring Network One-Way Transit Time," Research Technical Report ALR-2003-051, Avaya Labs.

Adler, M., and Mitzenmacher, M. (2001), "Towards Compressing Web Graphs," in *Proceedings of the IEEE Data Compression Conference*, Snowbird, UT, pp. 203–212.

Barlow, R. E., Bartholomew, D. J., Bremner, J. M., and Brunk, H. D. (1972), *Statistical Inference Under Order Restrictions*, New York: Wiley, p. 13.

Basseville, M., and Nikiforov, I. V. (1993), *Detection of Abrupt Changes: Theory and Applications*, Englewood Cliffs, NJ: Prentice-Hall.

Bearden, M., Denby, L., Karacali, B., Meloche, J., and Stott, D. T. (2002a), "Assessing Network Readiness for IP Telephony," in *Proceedings of the 2002 IEEE International Conference on Communications (ICC-02)*, pp. 2568–2572.

——— (2002b), "Experiences With Evaluating Network QoS for IP Telephony," in *Proceedings of the Tenth International Workshop on Quality of Service (IWQoS 2002)*, pp. 259–268.

Bestavros, A., Byers, J., and Harfoush, K. (2002), "Inference and Labeling of Metric-Induced Network Topologies," in *Proceedings of the IEEE Infocom*, New York, NY, pp. 628–637.

Buja, A., and Swayne, D. (2002), "Visualization Methodology for Multidimensional Scaling," *Journal of Classification*, 19, 7–43.

Cáceres, R., Duffield, N. G., Horowitz, J., and Towsley, D. (1999), "Multicast-Based Inference of Network Internal Loss Characteristics," *IEEE Transactions on Information Theory*, 45, 2462–2480.

Castro, R., Coates, M., Liang, G., Nowak, R., and Yu, B. (2004a), "Network Tomography: Recent Developments," *Statistical Science*, 19, 499–517.

Castro, R., Tsang, Y., and Nowak, R. (2004b), "Likelihood-Based Hierarchical Clustering," *IEEE Transactions on Signal Processing*, 52, 2308–2321.

Chen, A., Cao, J., and Bu, T. (2007), "Network Tomography: Identifiability and Fourier Domain Estimation," in *IEEE Infocom*.

Claffy, K. C., Polyzos, G. C., and Braun, H. W. (1993), "Application of Sampling Methodologies to Network Traffic Characterization," in *Proceedings the ACM SIGCOMM*, pp. 13–17.

Coates, M., Castro, R., Nowak, R., Gadhiok, M., King R., and Tsang, Y. (2002), "Maximum Likelihood Topology Identification From Edge-Based Unicast Measurements," in *Proceedings of the ACM Conference on Sigmetrics*, Marina del Rey, CA, pp. 11–20.

Cortes, C., Pregibon, D., and Volinsky, C. (2003), "Computational Methods for Dynamic Graphs," *Journal of Computational and Graphical Statistics*, 12, 950–970.

DallAsta, L., Alvarez-Hamelin, I., Barrat, A., Vazquez, A., and Vespignani, A. (2006), "Exploring Networks With Traceroute-Like Probes: Theory and Simulations," *Theoretical Computer Science*, 355, 6–24.

de Goyan, J.-M. (1998), available at *http://tldp.org/HOWTO/Multicast-HOWTO.html*.

Duffield, N. G. (2004), "Sampling for Passive Internet Measurement: A Review," *Statistical Science*, 19, 472–498.

Duffield, N. G., and Lo Presti, F. (2004), "Network Tomography From Measured End-to-End Delay Covariance," *IEEE/ACM Transactions on Networking*, 12, 978–992.

Duffield, N. G., Horowitz, J., Lo Presti, F., and Towsley, D. (2002), "Multicast Topology Inference From Measured End-to-End Loss," *IEEE Transactions on Information Theory*, 48, 26–45.

Duffield, N. G., Lund, C., and Thorup, M. (2005), "Estimating Flow Distributions From Sampled Flow Statistics," *IEEE/ACM Transactions on Networking*, 13, 325–336.

Eppstein, D., Galil, Z., and Italiano, G. F. (1999), "Dynamic Graph Algorithms," in *Algorithms and Theoretical Computing Handbook*, ed. M. J. Atallah, Boca Raton, FL: CRC Press, Chap. 8.

Gilbert, A. C., and Levchenko, K. (2004), "Compressing Network Graphs," in *Proceedings of the KDD Conference*, Seattle, WA.

ITU P.800.1 (2006), "Mean Opinion Score (MOS) Terminology," available at *http://www.itu.int/rec/T-REC-P.800.1-200607-I/en*.

Jeske, D. R., and Chakravartty, A. (2006), "Effectiveness of Bootstrap Correction in the Context of Clock Offset Estimators," *Technometrics*, 48, 530–538.

Jeske, D. R., and Sampath, A. (2003), "Estimation of Clock Offset Using Bootstrap Bias-Correction Techniques," *Technometrics*, 45, 256–226.

Karacali, B., Denby, L., and Meloche, J. (2004), "Scalable Network Assessment for IP Telephony," in *Proceedings of the 2004 IEEE International Conference on Communications (ICC-04)*, pp. 1505–1511.

Lawrence, E., Michailidis, G., and Nair, V. N. (2006a), "Network Delay Tomography Using Flexicast Experiments," *Journal of the Royal Statistical Society*, Ser. B, 68, 785–813.

——— (2007) "Statistical Inverse Problems in Active Network Tomography," in *Statistical Inverse Problems*, eds. R. Liu, W. Strawderman, and C. H. Zhang, Hayward, CA: IMS, to appear.

Lawrence, E., Michailidis, G., Nair, V. N., and Xi, B. W. (2006b), "Network Tomography: A Review and Recent Developments," in *Frontiers in Statistics*, eds. J. Fan and H. Koul, London: Imperial College Press, pp. 345–366.

Liang, G., and Yu, B. (2003), "Maximum Pseudo-Likelihood Estimation in Network Tomography," *IEEE Transactions on Signal Processing*, 512, 2043–2053.

Liang, G., Taft, N., and Yu, B. (2006), "A Fast Lightweight Approach to Origin-Destination IP Traffic Estimation Using Partial Measurements," *IEEE Transactions on Information Theory*, 52, 2634–2648.

Lo Presti, F., Duffield, N. G., Horowitz, J., and Towsley, D. (2002), "Multicast-Based Inference of Network-Internal Delay Distributions," *IEEE/ACM Transactions on Networking*, 10, 761–775.

Michailidis, G. (2006), "Data Visualization Through Their Graph Representations," in *Handbook of Computational Statistics: Data Visualization*, eds. C. Chen, W. Hardle, and A. Unwin, New York: Springer-Verlag, to appear.

Michailidis, G., Nair, V., and Xi, B. W. (2007), "Fast Estimation Methods for Estimation of Loss Rates in Active Network Tomography," preprint.

Moon, S. B., Skelly, P., and Towsley, D. (1999), "Estimation and Removal of Clock Skew From Network Delay Measurements," in *Proceedings of IEEE Infocom 1999*, New York, NY, pp. 227–234.

Paxson, V. (1997), "End-to-End Routing Behavior in the Internet," *IEEE/ACM Transactions on Networking*, 5, 601–615.

——— (1998), "On Calibrating Measurements of Packet Transit Times," in *Proceedings of the ACM Sigmetrics*, Madison, WI, pp. 11–21.

Peterson, L., and Davie, B. (2003), *Computer Networks: A Systems Approach*, San Francisco: Morgan Kaufmann.

Shih, M. F., and Hero, A. O. (2003), "Unicast-Based Inference of Network Link Delay Distributions With Finite Mixture Models," *IEEE Transactions on Signal Processing*, 51, 2219–2228.

Solka, J. L., Marchette, D. J., and Wallet, B. (2000), "Statistical Visualization Methods in Intrusion Detection," *Computing Science and Statistics*, 32, 16–24.

Stoumbos, Z., Reynolds, M. R., Ryan, T. P., and Woodall, W. H. (2000), "The State of Statistical Process Control as We Proceed Into the 21st Century," *Journal of the American Statistical Association*, 95, 992–998.

Tsang, Y., Coates, M., and Nowak, R. D. (2003), "Network Delay Tomography," *IEEE Transactions on Signal Processing*, 8, 2125–2135.

Vardi, Y. (1996), "Network Tomography: Estimating Source-Destination Traffic Intensities From Source Data," *Journal of the American Statistical Association*, 91, 365–377.

Xi, B. W., Michailidis, G., and Nair, V. (2006), "Estimating Network Loss Rates Using Active Tomography," *Journal of the American Statistical Association*, 101, 1430–1449.

Yang, L., and Michailidis, G. (2006), "Sample-Based Estimation of Network Traffic Flow Characteristics," in *Proceedings of Infocom 2007*, pp. 1775–1783.

Yang, X. (2006), "Design of Probing Experiments and Online Monitoring of Network Performance," unpublished doctoral dissertation, The University of Michigan.

Zhang, L., Liu, Z., and Xia, C. H. (2002), "Clock Synchronization Algorithms for Network Measurements," in *Proceedings of Infocom 2002*, New York, NY, pp. 160–169.

Zhang, Y., Roughan, M., Lund, C., and Donoho, D. (2003), "An Information-Theoretic Approach to Traffic Matrix Estimation," in *Proceedings of the ACM SIGCOMM*, Karlsruhe, Germany, pp. 206–217.