Assessing Network Readiness for IP Telephony

M. Bearden, L. Denby, B. Karaçalı, J. Meloche, D. T. Stott
Avaya Labs Research
233 Mount Airy Road
Basking Ridge, NJ 07920
{mbearden,ld,karacali,jmeloche,stott}@research.avayalabs.com

***Abstract*— Networked multimedia applications require stringent real-time QoS guarantees. Successful deployment of such applications closely depends on the performance of the underlying data network. The characteristics and the QoS requirements of these applications are different from traditional data applications. Hence, prior to deployment it is necessary to evaluate a network from a multimedia perspective. In this paper, we focus on IP Telephony and describe a framework for providing tools for IP Telephony readiness evaluation. This framework can be easily generalized to other multimedia applications. Our approach injects synthesized voice traffic and measures perceived end-to-end quality. We present a novel idea of relating voice quality metrics to the performance of data network devices. Following the proposed framework, we developed a prototype tool to evaluate a network and to identify problems, if any, prior to IP Telephony deployment. Our tool automatically discovers the topology of a given network, and collects and integrates network device performance and voice quality metrics. We describe the architecture of our tool and provide sample outputs from a network consisting of 129 devices.**

I. Introduction

Recent industry trends show that multimedia holds the potential for a wide range of applications. A significant challenge to making such potential applications widely available is ensuring that the IP network can provide a minimum quality of service (QoS) level. In this paper we focus on determining whether a network can handle applications with stringent QoS requirements. We specifically focus on IP Telephony, though our contributions can be generalized to other multimedia applications.

IP Telephony constitutes a challenge for data networks since it merges two application domains with different QoS standards, voice and data. While voice is very sensitive to delay and jitter, traditional data applications are more tolerant with respect to these performance metrics. Optimizing a high performance network for data traffic may adversely affect successful IP Telephony deployment. Recently, we developed a prototype tool called ExamiNet^{TM}, which addresses the problem of evaluating the ability of the network to deliver the quality of service necessary for IP Telephony deployment. Its purpose is to assist network engineers within Avaya Inc. at the pre-sales stages, prior to the installation of new IP Telephony equipment.

The authors thank the following people for their contributions to this work: Sachin Garg, Jenny Li, Mark Karol, Shane Sullivan, Clayton Whitehead, and Kenneth Kane.

The contributions of this paper are three-fold: First, we provide a framework for network assessment and IP Telephony readiness evaluation. Second, we describe our prototype tool, ExamiNet^{TM}, that automatically discovers the topology of a given network, collects and integrates network device status and voice statistics, and provides a mechanism to identify problems in the network. Third, we present a novel idea of relating voice quality metrics to the network devices. Our approach is to inject synthesized voice traffic in the network and to measure each endpoint's perspective of the end-to-end quality of service. At the same time, we monitor the load and utilization of the network devices that route synthesized voice packets. Our assumption is that the quality of service for these calls is a function of the utilization and load on the network devices that are on the call paths. We relate the network load on the call paths to voice quality parameters to identify network problems that may prevent an acceptable IP Telephony deployment.

An important aspect of our approach is that it does not rely on network and call simulation models. We use synthesized traffic injection and observe actual end-to-end QoS measurements, rather than relying on traffic *simulation* to predict the end-to-end QoS. In particular, we seek to observe call behavior at actual peak network load, rather than attempting to simulate and predict network load.

This paper is organized as follows: In Section II we provide a summary of existing commercial tools and related work. In Section III we present the preliminary information and the notation used throughout the paper. In Section IV we describe a framework for network assessment for IP Telephony. In Section V we provide the architecture for ExamiNet^{TM}. Our conclusions and future work are presented in Section VI.

II. Related Work

ExamiNet^{TM} supports a wide range of functionality in network evaluation. In this section we provide related work pertaining to various features of ExamiNet^{TM} starting with literature on network topology discovery. Then we list major commercial tools for monitoring network devices followed by relevant tools on voice traffic injection and voice quality monitoring. Finally we present tools that address integrating network device performance with voice call metrics.

A few examples of topology discovery at layer 3 are given in [1], [2], [3]. The approaches described in these papers mainly

focus on mapping the topology of the Internet backbone rather than that of an enterprise network. The work in [4] presents and compares ping-, traceroute-, and DNS-based techniques to obtain the layer-3 topology. Despite the importance of the layer-2 topology, little literature is available. One approach to generate the layer-2 topology between switches was presented in [5] and improved upon in [6]. This approach discovers the topology by processing the forwarding tables obtained from each switch via SNMP. Its accuracy depends on how complete forwarding tables are, and it has not been proven with VLANs. Some switch vendors (e.g., [7]) have tools for layer-2 topology discovery using proprietary MIB extensions on their own devices. A few commercial tools (e.g., [8], [9]) have recently added claims to support heterogeneous layer-2 topology discovery using proprietary techniques.

Many commercial tools for network performance monitoring and management are available. A few examples are Hewlett-Packard's HP Openview [10], Lucent's VitalSuite [11], Patrol DashBoard [12], Omegon's NetAlly [13], the Felix project from Telcordia Technologies [14]. In addition to these is the open source MRTG. These tools provide detailed network statistics, but they are limited in their ability to export the raw data collected to other tools for analysis purposes.

The emerging multimedia market has led to the development of new tools for testing the performance of multimedia applications. Among the commercial tools that support Voice over IP (VoIP) tests in their suite are NetAlly [13], Chariot [15], Hammer [16], and Explorer [17]. These tools differ in the way they inject voice traffic but they collect similar end-to-end measurements including delay, jitter, and packet loss.

Among the tools that are on the market, Agilent Technologies and Cisco Systems provide tools that address network assessment for IP Telephony. Agilent Technologies' suite of tools for testing IP Telephony services [18] consists of three main products, Voice Quality Tester (VQT), IP Telephony Analyzer, and IP Telephony Reporter. Voice Quality Tester measures voice quality objectively, without having human listeners. This system supports one-way and round-trip delay measurements, echo, and clarity (a measure of voice quality). IP Telephony Analyzer captures RTP packets and calculates various statistics for each RTP stream such as, packet loss, and jitter. Additionally, for each connection and protocol, it collects statistics on the number of frames, bytes, and frame errors, and the utilization. IP Telephony Reporter merges the call quality statistics provided by VQT and the packet-network statistics provided by the IP Telephony Analyzer by importing result files from both of the components.

Cisco Systems provides a solution called VoIP Readiness Net Audit for network assessment [19]. It uses proprietary SNMP-based tools for data collection from network devices. The goal of this solution is to assess the general health of the network. The service focuses on performance analysis of routers and switches and delivers an executive report describing the overall network performance and predicted VoIP readiness but it does not integrate voice quality statistics with network device statistics.

Each of the commercial tools mentioned in this section provides a part of ExamiNet^{TM}'s functionality. None of these tools, however, integrate voice quality metrics to network device statistics on the voice path as closely as ExamiNet^{TM}. Simply using a variety of existing tools to accomplish such a close integration is non-trivial since each tool has different interfaces, data formats, and limited data import/export support. Another major obstacle for integration is that the granularity of time measurements is different for each tool. Very few commercial tools provide fine granulated measurements, i.e., monitoring in the order of seconds.

Network QoS monitoring and evaluation can be performed in an intrusive (e.g., [20]) or a non-intrusive (passive) way (e.g., [21]). For the purposes of ExamiNet^{TM}, RTP packet injection is necessary to determine the type of service the actual voice packets would receive. Furthermore, collecting statistics from switches and routers (e.g., via SNMP) introduces unavoidable traffic. There is extensive literature on efficient network monitoring and QoS evaluation for data networks (e.g., [22], [23], and [24]). These approaches employ statistical techniques to estimate network node and link characteristics. In our future work we will consider these techniques as part of our analysis efforts to assess network characteristics from a multimedia perspective.

III. Preliminaries

We represent the network topology as a graph, $G = (D, L)$ where D is a set of devices and L is a set of links. Each device in D is of type router or switch. Let D_i and D_j be two devices in D where $1 \leq i, j \leq |D|, i \neq j$. There is a link between D_i and D_j if and only if there is a direct communication path between D_i and D_j. $I_{i,j}$ denotes the jth interface of device D_i. Each link out of a device in the graph represents an interface in the network.

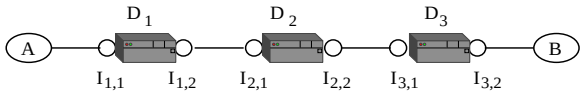

Fig. 1. Call Path

Figure 1 illustrates the network elements involved in a sample voice call between endpoints A and B. We refer to a device that can initiate and respond to voice calls as an endpoint. We assume the path from A to B is the reverse of the path from B to A. A line between two devices denotes a bidirectional link. Assume that the voice packets pass through three routers indicated as D_1, D_2, and D_3. The router interfaces that the packets traverse are also marked on the figure. All network elements that participate in a voice call form the call path. For the call shown in Figure 1, the call path consists of A, $I_{1,1}$, $I_{1,2}$, $I_{2,1}$, $I_{2,2}$, $I_{3,1}$, $I_{3,2}$, and B.

Our analysis requires objective and measurable metrics to quantify the quality of a voice call. We consider four IP performance metrics that affect the quality of IP Telephony calls: one-way delay, jitter, packet loss, and packet loss burst. One-way delay is defined as the time from when the source sends the packet until the time the destination receives it. To measure jitter we use a running average of the differences in inter-packet arrival times as described in [25]. Packet loss from a source to a destination during an interval refers to the ratio of the number of packets lost to the number of packets sent during that interval [25]. Packet loss burst during an interval refers to the maximum number of consecutive packets lost during that interval [26]. For Avaya equipment, Avaya Inc. standards recommend one-way delay in the network of less than 50 ms, jitter of less than 20 ms and data loss of less than 0.2% for voice calls to be considered acceptable [27]. A call is considered bad as soon as, at any point during the call, any of the quality metrics fails to meet these specifications.

In assessing call quality, user perception is the definitive metric. Relating low-level quality metrics such as end-to-end delay, jitter, packet loss, and packet loss burst, to the more subjective notion of user perception is important in understanding call quality. Recently, Mean Opinion Score (MOS) is defined in ITU recommendation P.800 for assessing voice call quality based on the subjective opinion of listeners. In order to relate MOS to objective metrics, ITU recommendation G.107 defines an objective score "R factor" that can be mapped to a MOS score. In our future work, we will address relating the low-level quality measurements to metrics that consider user perception.

IV. Framework

In this section we describe a framework for providing tools which makes it easier to assess the IP Telephony readiness of a network. This framework is based on our goal to relate end-to-end performance metrics to the load on network devices. Our assumption is that the only factor affecting the quality of a call is the performance of network devices on the call path.

Below, we show the steps in applying our framework:

- Topology Discovery Phase
- Network Device Monitoring Phase
- Call Synthesis & Call Quality Monitoring Phase
- Analysis Phase

Topology Discovery Phase

Discovering the network topology is the first step in our framework. This phase identifies the set of devices in the network, their function (e.g., router or switch), and their interconnections. Furthermore, it identifies the path between any two devices in the given network. The accuracy of our analysis depends on how well the network elements on the voice call path are identified. With the increased use of virtual LANs (VLANs), layer-2 switches are replacing layer-3 routers except those at the edge of the enterprise. With this paradigm shift, the layer-2 topology becomes more important. Because these switches introduce delay, jitter and packet loss, they cannot be ignored. For instance, in networks with VLANs, the layer-2 path may traverse many devices hidden at layer 3. In such situations, using layer-3 topology can be rather misleading.

Network Device Monitoring Phase

During the network device monitoring phase, network load statistics are collected from the discovered devices in the network. Various statistics may be used as an indication of load on a network device. For instance, for a given device, the number of incoming and outgoing octets on all of its interfaces, number of discarded packets on all of its interfaces, its CPU usage, etc. constitute measures of load.

Given that call quality can be affected by adverse network conditions even if they last but a short amount of time, it is necessary to collect the device statistics at an adequately fine resolution.

Call Synthesis & Call Quality Monitoring Phase

Call synthesis is carried out concurrently with the network device monitoring. At the time the voice traffic is injected, two types of information are collected: call quality metrics and layer-3 path information. Call quality metrics of interest are end-to-end delay, jitter, packet loss, and packet loss burst. The layer-3 path is crucial because our analysis depends on it.

Analysis Phase

After the collection of an adequate amount of call and load statistics, the analysis phase begins with the integration of call and network device load statistics. The timestamps at which measurements are collected are critical in this integration. An important issue from the perspective of network assessment is to identify the "bad" calls and the network devices on the call paths of these bad calls in an effort to determine the root cause of problems. Identifying such problems requires the expertise of a network engineer. The large amount of information collected must be analyzed in a systematic way, leading to the isolation of any problematic devices. Heuristics to draw attention to problematic network devices facilitate the analysis.

V. Prototype Tool Architecture

In this section we describe ExamiNet^{TM} and provide details of how we implemented the framework outlined in the previous section. Figure 2 shows our architecture. The direction of arrows correspond to the direction of information flow.

Our prototype tool consists of five main components, namely, topology discovery, network device monitoring, call generation & call quality monitoring, database, and visualization & analysis. In the following sections we describe each component.

A. Network Topology Discovery

The ability to discover devices in a customer's network and their topology at both layer 3 and layer 2 using SNMP queries is key to ExamiNet^{TM}. Network topology discovery serves three important purposes. First, it identifies devices in the network to monitor. Second, it allows the endpoint selection process to intelligently choose endpoint pairs whose paths cover the entire

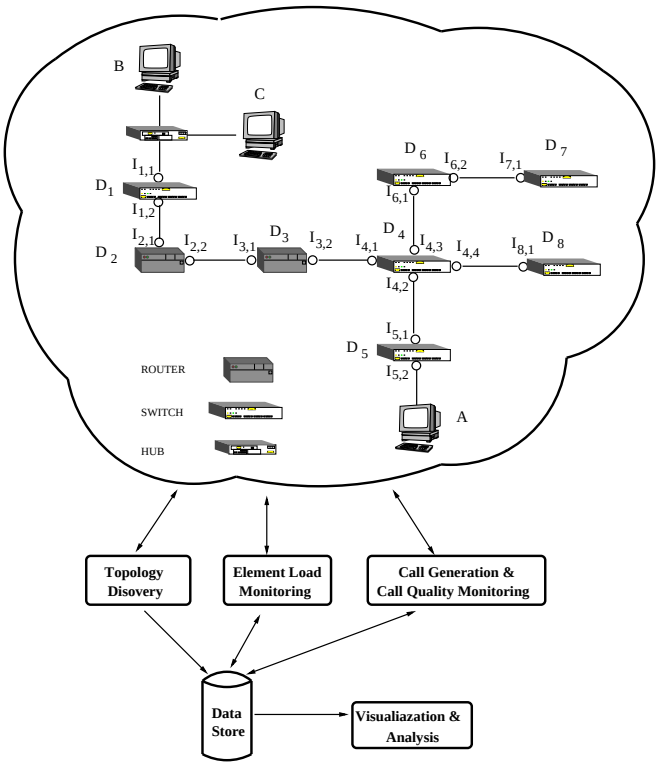

Fig. 2. Prototype Tool Architecture

network. Third, it identifies which network elements are in the path between endpoint pairs so that the analysis can associate call data between the endpoints and data collected from network devices.

Field engineers have found that few customers can accurately describe their network topology. Hence, instead of relying on the customer we must generate this information ourselves. We assume, however, that the customer will provide us with the IP address ranges used at the sites.

The procedure for using SNMP to collect the network topology has several steps. The results from this procedure are written to the database:

1) **Probe:** This step probes the network for devices by sending SNMP queries to every IP address at the site(s).
2) **Classify:** For each address responding to the probe, this step sends specific queries to determine the device type (e.g., router, switch, and printer). The classification process first tries to lookup the type based on the system OID. If the system OID lookup fails, the process classifies the device using a heuristic based on MIB variables such as system.sysServices, ip.ipForwards, and dot1d.bridgeType.
3) **Identify Aliased Addresses:** Some devices respond to multiple IP addresses (e.g., multi-honed routers). This step finds addresses belonging to the same device by assigning an ID that is unique to each device based on the physical addresses in the interface table. Responses from different IP addresses with the same ID must belong to the same device.
4) **Collect Router Tables:** For each router, this step collects the interface table, route table, IP address table, and MAC_address_to_IP_address table based on interfaces.ifTable, ip.ipRouteTable, ip.ipAddrTable, and ip.netToMediaTable MIB tables, respectively.
5) **Remove Entries at Subnet Boundaries:** Some devices respond to queries to their network address or broadcast address, which can be found in the routers' route tables. This step removes these entries from the results of previous steps.
6) **Collect Switch Tables:** For each switch, this step collects the interface table (interfaces.ifTable), forwarding table (dot1dBridge.dot1dTp.dot1dTpFdbTable) which maps physical address to port number, and tables needed to generate the spanning tree from the Bridge MIB [28]. Some switches require vendor-specific MIBs to generate these tables.

The topology information can be used later to find paths between endpoints at layer 3 or at layer 2. The path between routers can be found directly by looking up the destination address in the current router's route table to find the address of the next router. One difficulty is finding the default router for each endpoint. We consider the default router to be the the router with an entry in the route tables that (a) uses the smallest network mask of the entries including the subnet, and (b) has a next hop address in the same subnet. Should multiple routes fit these criteria, a heuristic (namely selecting the one with the lowest IP address) is used. The path will be incomplete if it includes an undiscovered device. In this case, the procedure finds segments from each endpoint to the first undiscovered device. These undiscovered devices are identified as an "unknown router cloud."

The layer-2 path can be found between any IP addresses in the same subnet (i.e., between any two devices in the layer-3 path). As with the layer-3 path, the part of the path between network devices is simple–the spanning tree defines the path. Finding the first switch in the path is more difficult ([5] explains complexities of this problem). Our approach uses the switch's forwarding tables, the spanning tree, and the router's MAC table, all of which are stored in the database. First, it looks up the endpoint's MAC address from router's MAC table. All switches, except the one closest to the endpoint, will forward packets to the MAC address along a trunk line (links between switches are specified in the spanning tree). Assuming the endpoint's MAC address is in an entry in the switch's forwarding table, it will be the only such entry that uses an interface that is not part of a trunk line.

VLANs are becoming a popular way to integrate several independent logical switched networks on a single physical network. VLANs affect the layer-2 topology dramatically, but have no impact at layer 3. The path through the network is a function of the VLAN since each VLAN on the same physi-

cal network often has its own spanning tree. The component to generate the layer-2 path needs to use the VLAN ID as a parameter. This potentially complicates visualizing the network traffic because the entire topology becomes a graph rather than a tree. Most VLAN implementations use vendor-specific MIBs for layer-2 tables; code for each vendor-specific MIB must be generated individually.

One consideration is how to deal with non-SNMP devices (those devices where we cannot collect SNMP data). Though most network devices are capable of using SNMP, it is inevitable that we encounter situations where we cannot access SNMP data from the device (e.g., SNMP may be disabled or never configured, access lists may block access, or unconventional community strings may be used). ExamiNet^{TM} attempts to provide some useful information even when some non-SNMP devices are encountered. For a path that goes through a non-SNMP device, our primary goal is to determine the interfaces on SNMP enabled devices on the path. At layer 3, the IP address of the first non-SNMP device on a path (in each direction) is also known from its neighbor's route table.

Another consideration is how to deal with virtual ports. To obtain a layer-3 topology, we would like to model each link as a pair of interfaces on two connected devices. Then the interface indices would be used to find the interface-specific statistics in the interface table (i.e., interface.ifTable). For most traditional routers, this approach works fine. For newer devices, however, such as layer-2/3 switches (e.g., a Cajun switch, which is a switch with a layer-3 routing module in the same box) traffic to a particular router IP address can use any physical switch port (depending on the layer-2 topology and VLAN). These devices generally use a virtual interface for the layer-3 interface, yet they only give interface statistics for the physical interfaces. Thus, care must be taken to identify virtual ports when analyzing layer-3 data. The complete layer-2 topology gives the physical port from the router to the next-hop address.

B. Network Device Monitoring

The network device monitoring component of ExamiNet^{TM} collects network utilization and load statistics on discovered devices via SNMP MIBs. The database provides the list of devices (from the discovery phase) to be monitored.

This component collects values for two types of MIB variables from the SNMP agents on discovered devices at regular intervals. The first type is device-specific MIB variables such as the total number of input datagrams received on all interfaces. The second type is interface-specific MIB variables such as the total number of octets received on an interface.

Network elements are polled at regular intervals to obtain the values of sets of MIB variables. All MIB variables in a set are of the same type. Each set is associated with a polling interval and some sets are polled more frequently than others. For example, the set of MIB variables storing the total number of octets received and sent has a more frequent polling interval. The basis for more frequent polling is that these MIB variables are a direct indication of network traffic and their values change very fast. The raw counter values of the MIB variables are stored in the database.

One issue in monitoring is that we are introducing SNMP traffic in the network and load on network devices by querying their MIBs. So far, our tests during each use of ExamiNet^{TM} indicate that the impact of SNMP traffic on the monitored devices is insignificant. Furthermore, ExamiNet^{TM} provides a real-time estimate of the response time of each device during collection periods.

C. Call Generation & Monitoring

The call generation and monitoring component of ExamiNet^{TM} injects voice traffic to the network while collecting call quality metrics and layer-3 path information. Call quality metrics are measured at the endpoints of each call, and measurements are preserved for both directions of RTP traffic flow. The layer-3 path information, collected using traceroutes initiated by the endpoints during the synthesized call, is used to verify that the call is following the path predicted by the discovery phase based on router tables.

Voice traffic injection is carried out "around the clock" for several days, typically at least five business days. The objective is to ensure that the data collection occurs during time sensitive congestion that may occur in the network, and in particular to observe the network at daily and weekly peak, or "busy hour", loads.

In general, there are several ways to generate the voice traffic, including using actual or simulated IP telephony equipment. The basic requirement is that the injected traffic should emulate a full duplex call. In other words, RTP packets should be exchanged at regular intervals between a pair of call endpoints. The call endpoints can be IP telephones, computing devices that simulate the RTP packet flows from telephones, or a combination of both. For the sake of simplicity, we refer to the generated voice traffic as synthesized traffic regardless of how the voice traffic is generated.

Voice traffic injection has many parameters that impact the effectiveness of our approach. These include:

- where call endpoints are placed in the network, both physically with respect to network devices and logically with respect to VLANs,
- what subset of the possible endpoint pairs will be used to synthesize calls and how many calls should be occurring concurrently,
- what call duration and inter-call intervals to use,
- which standard codecs should be simulated when generating RTP packets, and
- what ports and QoS markings (TOS, Diffserv, VLAN tags) to use for generated RTP packets.

Placement of call endpoints directly affects which part of the network is traversed by the call traffic. In order to draw conclusions about a network, injected voice traffic should cover

the entire network. Note that covering the entire network is not necessarily sufficient for the purpose of identifying problematic parts of a network. It is also necessary to be able to distinguish the effect of each hop on call quality. Call duration and inter-call intervals for synthesized traffic should reflect respective parameters for actual voice calls. Selection of codec impacts the payload size and the packet transmission rate. QoS markings affect the way network devices handle voice packets. Our approach requires that the synthesized traffic receive the same treatment as the actual voice packets after the deployment of IP telephony equipment.

The eventual analysis of the collected measurements should support the observation of all potential end-to-end QoS problems, and also support root cause analysis of identifying how the different network elements affect end-to-end QoS. Thus prior to the onset of voice traffic injection, the call pattern must be planned appropriately, using an awareness of the network topology, to provide the needed network coverage.

The call generation and monitoring component of ExamiNet^{TM} has a separate user interface that allows the user to specify a sequence of calls called a *call pattern.* Each call is identified by a pair of endpoints, QoS setting, codec/payload, packet rate, port number, jitter buffer size, start time, and duration. A single endpoint may be specified to appear any desired number of times within a given call pattern.

During the call generation phase, a call control module automates the initiation of calls and collection of QoS statistics. Endpoint software must be installed on a computer to send and receive synthesized traffic and to collect and report statistics to the call control module. Let E_1 and E_2 be two endpoints in the network running the endpoint software. To initiate a synthesized call between E_1 and E_2 at time t, the call control module sends control information, including call parameters, at time t to the control agents running on both E_1 and E_2. E_1 and E_2 execute the calls and report call statistics back to the call control module. The call control module stores the call statistics such as delay, jitter, and packet loss in the database. Current implementation uses a commercial tool [15] for the endpoints.

Call patterns are currently generated using intuitive heuristics. Our current algorithm relies on randomly distributing the endpoints but also ensures that endpoint pairs separated by long paths are exercised. The motivation for ensuring the inclusion of long paths is that we would like to determine the worst possible delay that voice traffic would incur in the network. Without any prior information on a network, paths that have more hops are likely to have more delay. Furthermore, using long paths increases our chances of encountering problems in the network early on. Many variations on this strategy are possible. We are working on developing algorithms for call pattern generation and quantifying their network coverage.

D. Database

The ExamiNet^{TM} architecture stores all the information collected from the network in a persistent store (currently a MySql database). The other components interact with the database. The discovery component stores the network topology and device data in the database. The element monitoring component stores network load measurements at each polling interval. Similarly, call generation and call quality monitoring component stores call quality statistics at well defined intervals.

It is critical for integration purposes that the measurements provided by the network device monitoring component and the call quality monitoring component use the same reference clock. Each measurement provided by either of these components has a timestamp that is used when integrating measurements from different sources. Currently different components of ExamiNet^{TM} may execute on different hosts and send their data to the database. In such cases, we use NTP service for clock synchronization to ensure that measurements are marked according to synchronized clocks.

Database storage requirements are a concern for long monitoring periods. One of our preliminary empirical studies ran for approximately 4 days and generated a database size of 1.5 GB. This study involved 10 routers and switches with an average of 97 interfaces per device. The monitor used 60 second polling interval for 2 MIB variable sets of 11 and 18 MIB variables and 10 second polling interval for 1 MIB variable set of 2MIB variables. For the overall duration of 4 days, 18 endpoints participated in calls, with an average of 2541 calls each.

E. Visualization and Analysis

ExamiNet^{TM} provides a tabular and an interactive interface to the information stored in the database. In this section we describe these interfaces and provide example outputs from ExamiNet^{TM}.

Tabular Interface

Load statistics on monitored devices are accessible through a web page. Users may select a device or an interface and specify a time frame. Based on user specification, the visualization component queries the database and returns a table that contains the values of all MIB variables matching the specification. Each column of this table corresponds to the raw counter values for a distinct MIB variable collected over the specified time frame for the specified device/interface. Each row contains counter values for all MIB variable statistics collected in one polling interval for the specified device/interface. The tabular interface allows exporting data to other applications.

Interactive Interface

The interactive interface displays a graphical representation of the network topology and uses it as an access mechanism to see various views. Figure 3 shows an ExamiNet^{TM} screenshot. When users select a network from the main window of the interactive interface, the network topology and a list of all call pairs for the selected network are available in separate windows. Figure 3 shows the network topology for a sample network recently evaluated by ExamiNet^{TM}. Users may select a device from the topology window and view detailed information on the selected device. The device details window in Figure 3 corresponds to

one of the routers on the network topology. This window contains device-specific information and a list of all calls going through it.

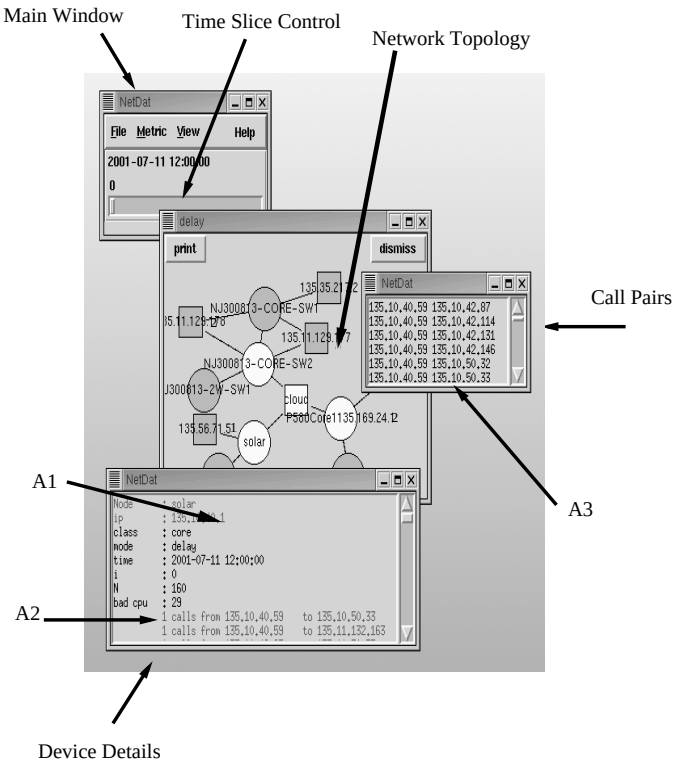

Fig. 3. ExamiNet^{TM} Screenshot

The interactive interface supports three types of views: device statistics view, call quality statistics view, and path statistics view. Each view contains a set of relevant plots.

The device statistics view displays plots of device- or interface-specific MIB variable statistics over time. For example, if a device and MIB variable ip.ipInReceives is selected, ExamiNet^{TM} displays a plot of the number of datagrams received per second by its interfaces. Similarly, if an interface and the inOctets metric are selected, ExamiNet^{TM} displays the plot of the number of bits per second it received. This view is accessible by clicking on the area marked as A1 in Figure 3.

The call quality statistics view consists of four plots pertaining to a user-specified call. These plots display one-way delay, jitter, packet loss, and packet lost burst of each direction of the call over time. All the plots in the view are in the same time scale. This view is accessible by clicking on the area marked as A2 and A3 in Figure 3.

The call path statistics view shows device-specific information for each device or interface on the call path. This view has two columns where each column corresponds to one way of the call. Each column consists of a sequence of plots, where each plot displays device-specific load information pertaining to a device on the call path. This view is accessible by clicking on the area marked as A2 and A3 in Figure 3.

The interactive interface provides heuristics to display interesting information, i.e., cases where low call quality is observed and further investigation is necessary to determine the cause. To identify such cases, the analysis and visualization component computes a status indicator for each device, time slice, and QoS metric. The time slice is a user-specified parameter and its granularity is adjustable. The possible QoS metrics are end-to-end delay, jitter, packet loss, and packet loss burst. ExamiNet^{TM} uses a coloring scheme to display the status of each device for various time slices and various QoS metrics. Color varies from red (unacceptable) to white (acceptable). For example, if all calls going through a device experience high delay during a time slice, that device is colored red for that time slice. Users may then study various views pertaining to the device in efforts to determine the cause of low call quality. For example, each device shown in Figure 3 has a color indicating its "delay" status at the specified time interval. The darker colors indicate high values of delay. Quality metrics other than delay may be selected from the main window of the interactive interface.

Example

The sample network shown in Figure 2 is part of a larger network consisting of 129 devices. ExamiNet^{TM} discovered this network and collected data for approximately four days. In this section we consider an example call between endpoints A and B shown in Figure 2. Figures 4 and 5 show the call quality statistics view and the call path statistics view for this sample call. The date markings on the x-axis have the format "month/day". During the experiment, host C shown in Figure 2 introduced FTP traffic to the call path from A to B. Both B and C are connected to switch D_1 through the same hub. As a result, the effect of this traffic was observed on interface $I_{1,1}$ as well as on some of the other devices on the call path. It is expected that this FTP traffic affects the call quality adversely. The shaded periods in Figures 4 and 5 correspond to the period of FTP traffic.

Figure 4 shows a sample call quality statistics view for the example call. Except during the FTP transfer, all low-level quality measurements (jitter, delay, packet loss, burst packet loss) meet the acceptable standards discussed in Section III. The plots in the view are in the same time scale and the dashed vertical lines correspond to one day.

Figure 5 shows the in bits per second and the out bits per second on the interfaces of the example call. Note that when octets received in interface $I_{i,j}$ equals octets sent from interface $I_{k,l}$, the view shows only octets sent from $I_{k,l}$. Each column corresponds to one way of the call. The speed of each interface in bits per seconds is marked on the upper left corner of each plot and must be taken into consideration when interpreting the utilization levels. The plots in the view are in the same time scale and the dashed vertical lines correspond to one day. Note that the time frame for the plots in this view is the same as those in Figure 4.

Introduction of FTP traffic through C was a small test to determine whether we could detect the impact of this traffic on

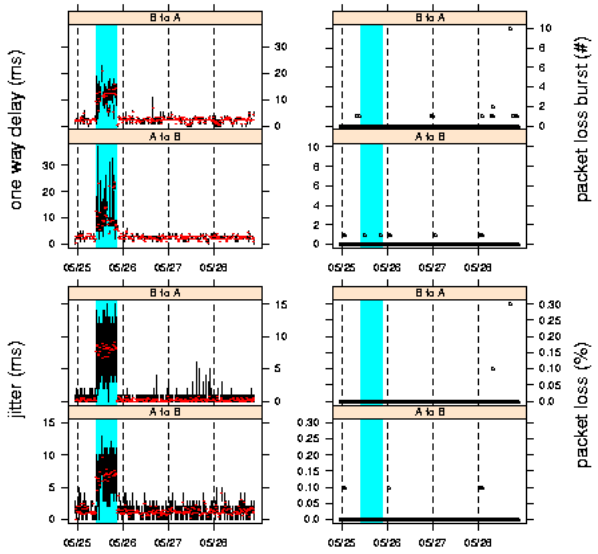

Fig. 4. Call Quality Statistics View

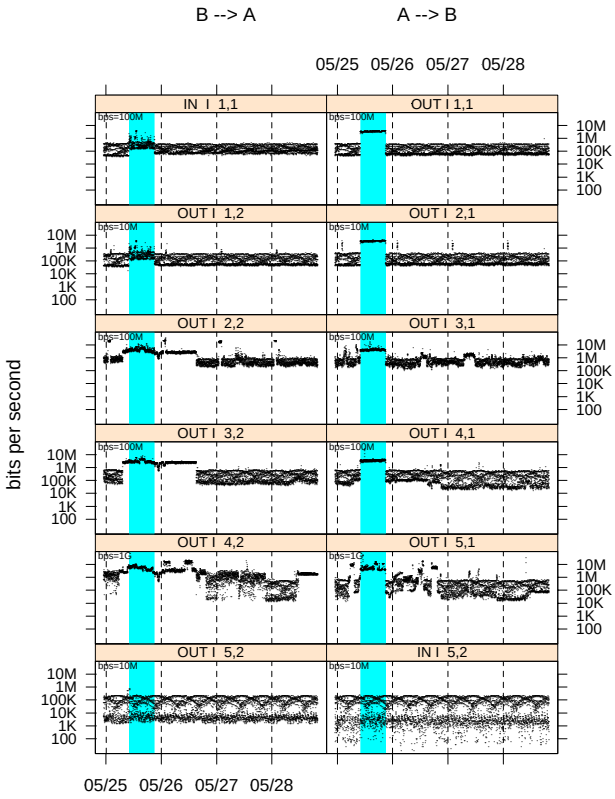

Fig. 5. Call Path Statistics View

the call and on the devices via ExamiNet^{TM}. As seen on both the call details and the call path details views, ExamiNet^{TM} captures the effects of this traffic.

VI. Conclusions & Future Work

In this paper we present a framework for providing tools which facilitate the assessment of IP Telephony readiness of a network. Though the framework is intended for IP Telephony, it can be easily generalized to other multimedia applications. This framework is based on our goal to relate end-to-end quality of service metrics to the load on network devices. Automatic network discovery is the first step in applying our framework. Upon completion of discovery, network device monitoring, injection of synthesized traffic, and call quality monitoring are carried out concurrently. At the end of the data collection period, which spans periods of varying network utilization levels, the analysis step begins. This step integrates all collected data and provides a mechanism to assess the performance of the network under consideration.

Recently, we developed a tool called ExamiNet^{TM} that implements our proposed framework. We describe the architecture of ExamiNet^{TM} and present sample outputs from ExamiNet^{TM}'s visualization and analysis component.

Our approach is based on a novel idea of relating voice quality metrics for a particular call path to the load on network devices on the call path. ExamiNet^{TM}'s discovery component identifies layer-2 paths between the endpoints in the network. These paths are essential in relating end-to-end call quality to the load on the network devices on the call path.

Our approach provides support for injecting synthesized traffic according to user defined parameters such as endpoint pair, codec, call length, call start and end time. These parameters define a call pattern. How well a specified call pattern covers a given network is a problem by itself. ExamiNet^{TM} currently selects call patterns based on intuitive heuristics such as random selection of call paths while ensuring the selection of long call paths (in efforts to observe "bad" calls fast). We are working toward defining some network coverage criteria and implementing algorithms that generate call patterns which satisfy a specified criteria.

Poor call quality characteristics may be due to a variety of factors, including a misconfigured networking element, an overloaded link, and improper prioritization for voice traffic. Given that individual calls are channeled through numerous elements and links, the reason for poor call performance is typically not easily determined. At the same time, a problem at some location in the network is likely to affect the performance of any call whose path goes through that location.

It is clear that the problem of correctly attributing the blame for poor performance is crucial to any diagnostic effort. We are currently developing models that take into account call performance metrics, network device monitoring time series and network topology to best determine the location and nature of underlying problems. The models can be fitted in the course of the project, not only making it possible to identify problems early on but also making it possible to change the focus of the measurement process to areas where there is greater uncertainty.

REFERENCES

[1] B. Huffaker, M. Fomenkov, D. Moore, and K. Claffy, "Macroscopic analyses of the infrastructure: Measurement and visualization of Internet connectivity and performance," in *Proc. of PAM2001–A workshop on Passive and Active Measurements*, Amsterdam, Netherlands, Apr. 23-24 2001.

[2] Hal Burch and Bill Cheswick, "Mapping the Internet," *IEEE Computer*, vol. 32, no. 4, pp. 97–98, Apr. 1999.

[3] Ramesh Govindan and Hongsuda Tangmunarunkit, "Heuristics for Internet map discovery," in *Proc. of the 2000 IEEE Computer and Communications Societies Conf. on Computer Communications (INFOCOM-00)*, Los Alamitos, CA, Mar. 26–30 2000, IEEE.

[4] R. Siamwalla, R. Sharma, and S. Keshav, "Discovering Internet topology," http://www.cs.cornell.edu/skeshav/papers/discovery.pdf, 1999.

[5] Yuri Breitbart, Minos Garofalakis, Cliff Martin, Rajeev Rastogi, S. Seshadri, and Abraham Silberschatz, "Topology discovery in heterogeneous IP networks," in *Proc. of the 2000 IEEE Computer and Communications Societies Conf. on Computer Communications (INFOCOM-00)*, Los Alamitos, CA, Mar. 26–30 2000, pp. 265–274, IEEE.

[6] Bruce Lowekamp, David R. O'Hallaron, and Thomas R. Gross, "Topology discovery for large ethernet networks," in *ACM SIGCOMM 2001*, San Diego, CA, Aug. 27–31 2001, pp. 237–248, ACM.

[7] Hewlett-Packard Co., *HP TopTools 5.5 User Guide*, 2001.

[8] Hewlett-Packard Co., "Discovering & mapping level 2 devices," http://www.openview.hp.com/library/papers/index.asp, 2001.

[9] Peregrine Systems, Inc., "InfraTools network discovery," http://www.peregrine.com, 2001.

[10] Hewlett-Packard Co., "HP OpenView," http://www.openview.hp.com.

[11] Lucent Technologies, "VitalQIP," http://www.qip.lucent.com.

[12] Bmcsoftware, "PATROL DashBoard," http://www.bmc.com/.

[13] Omegon, "White Paper: NetAlly," http://www.omegon.com/whitepapers.asp.

[14] C. Huitema and M. W. Garrett, "Project felix: Independent monitoring for network survivability," http://govt.argreenhouse.com/felix.

[15] NetIQ, *Chariot*, http://www.netiq.com/products/chr.

[16] Empirix, "Test and Monitoring Solutions for Web, Voice, and Network Applications," http://www.empirix.com/Empirix.

[17] Sunrise Telecom, *VoIP Explorer*, http://www.sunrisetelecom.com/voip/voiphome.shtml.

[18] Agilent Technologies, "IP telephony reporter," http://onenetworks.comms.agilent.com/agilentadvisor/J5422A.asp, 2001.

[19] Cisco Systems, "Cisco VoIP readiness net audit," http://www.cisco.com/.

[20] P. Skelly and M. Li, "EIPMon: an enhanced IP network monitoring tool," in *Proceedings from the IEEE Workshop on Internet Applications*, 1999, pp. 20 –27.

[21] I. Cozzani and S. Giordano, "A passive test and measurement system: Traffic sampling for QoS evaluation," in *Proceedings from the Global Telecommunications Conference*, 1998, vol. 2, pp. 1236 –1241.

[22] A.B. Downey, "Using pathchar to estimate Internet link characteristics," in *Proceedings of ACM SIGCOMM '99 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 1999, pp. 241–250.

[23] Yuri Breitbart, Minos Garofalakis, Cliff Martin, Rajeev Rastogi, S. Seshadri, and Abraham Silberschatz, "Efficiently monitoring bandwidth and latency in IP networks," in *Proc. of the 2001 IEEE Computer and Communications Societies Conf. on Computer Communications (INFOCOM-01)*, 2001, vol. 2, pp. 933 –942.

[24] R. Caćeres, N.G. Duffield, J. Horowitz, D. Towlsey, and T. Bu, "Multicast-based inference of network-internal characteristics: Accuracy of packet loss estimation," in *Proceedings from the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, 1999, vol. 1, pp. 371 –379.

[25] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time applications," http://www.ietf.org/rfc/rfc1889.txt, Jan. 1996.

[26] J.Q. Walker, "A handbook for successful VoIP deployment: Network testing, QoS, and more," http://www.netiq.com/ products/chr/whitepapers.asp.

[27] Avaya Inc., "Avaya IP voice quality network requirements," http://www1.avaya.com/enterprise/whitepapers, 2001.

[28] E. Decker, P. Langille, A. Rijsinghani, and K. McCloghrie, "Definitions of managed objects for bridges," RFC 1493, July 1993.