

# Simple On-Demand Overlays for Reliable Real-Time Traffic in Enterprise Networks

Bengi Karacali, Mark Karol, A.S. Krishnakumar, P. Krishnan and Jean Meloche

Avaya Labs

233 Mt. Airy Road, Basking Ridge, NJ 07920, USA

Email: {karacali,mk,ask,pk,jmeloche}@avaya.com

**Abstract**—Real-time applications such as IP Telephony demand strict quality of service (QoS) from the underlying network. As more enterprises deploy IP Telephony, it is becoming imperative to significantly improve the network reliability as seen by these applications. As such, we recently proposed a practical overlay architecture, ORBIT (Overlays providing Reliability for Better IP Telephony). Our architecture consists of intelligent endpoints and independent relays. The endpoints rapidly detect network QoS degradation and find alternate application layer paths (mid-call) to the destination that avoid the problems. The overlay is activated only on-demand and the system is easy to deploy. In this paper, we compare two measurement techniques, describe our initial prototype and provide empirical results of the prototype’s reaction time to network problems that indicate rapid response to QoS degradation.

**Keywords**—Overlay Networks; IP Telephony; Mid-call Redirection.

## I. INTRODUCTION

Real-time applications, especially IP Telephony, demand underlying infrastructures to both be reliable and provide acceptable Quality-of-Service (QoS). The networks supporting these traffic classes need to provide low packet delays, jitter, and losses, and this is often achieved through network design and by prioritizing such traffic. However, these techniques alone are not sufficient. For real-time applications, it is important to find ways to detect and avoid network failures and degraded QoS within seconds rather than the minutes it might take a network to recover. In enterprise networks, any reliability solution must also deal with several interesting challenges and practical constraints. It should be simple and, if possible, reuse existing enterprise components. In addition, it should conserve bandwidth whenever possible and should allow for incremental deployment.

In contrast with some prior IP reliability solutions [1], [2], we assume that applications can be modified to exploit network redundancy. Also, some solutions (e.g., [3]) concentrate on the possible benefits of multi-homing and not on the recovery time afforded by the technique, which is of importance in our IP Telephony work when we deal with mid-call redirection of calls.

Other prior overlay-based solutions [2], [4], [5], [6] concentrate on ISP-based networks rather than enterprise networks. They quote best recovery times on the order of tens of seconds, which while excellent for the applications they were targeting, are too large for mid-call redirection of IP Telephony. A key

difference with prior work is that prior work typically uses continual monitoring, rather than the on-demand approach that we present in this paper. Continual monitoring introduces complexity that is traditionally shunned by enterprises. It also limits the number of nodes that can be part of the overlay, which would be a detriment in enterprise IP Telephony where there are several tens of thousands of endpoints in large deployments. Furthermore, placing overlay nodes/relays inside the ISP core [6] is not applicable in the enterprise context. Finally, deployability is not addressed in the prior work.

In this paper, we present our application overlay architecture that we refer to as ORBIT (Overlays providing Reliability for Better IP Telephony) [7], [8]. The two key components of the ORBIT architecture are: *intelligent endpoints* and *independent relays* (which are not aware of each other’s presence and can be incrementally deployed). To simplify the management of the overlay network and avoid the complexity of an overlay routing protocol, ORBIT restricts the overlay network topology to one where there is *at most one intermediate relay node*. This architectural “restriction,” while seemingly minor, leads to major simplification in the measurement and decision making techniques and presents huge deployment and management cost benefits, without sacrificing on resiliency benefits. Our earlier work has shown that single-hop application relays can exploit many of the redundancies in enterprise networks [7], which is consistent with the observations for ISP networks [5], [6]. Furthermore, our earlier work has shown that a small number of relays is sufficient to protect call quality from single point failures [7].

Another important contribution of ORBIT is to place intelligence (essentially, additional software) in the endpoints. Since the endpoint is *always* involved in the conversation, it can rapidly detect degrading quality and *then* decide to monitor available overlay paths for traffic redirection. This has the desirable side effect of creating *on-demand overlays*, i.e., overlays that do not get activated unless there is a problem.

In this paper, we also describe an ORBIT prototype that we have built. Our initial experimental results with this prototype show that an endpoint can perform mid-call redirection of an IP Telephony call in the sub-second to seconds range, allowing the end-user to notice nothing more than a brief glitch in the stream. Most of the techniques in this paper are applicable to non-enterprise (e.g., ISP) networks and other real-time traffic (e.g., video). However, the presentation in this paper and the

reported experiments focus on IP Telephony in enterprises.

The rest of the paper is organized as follows. In Section II, we present the ORBIT architecture. In Section III, we elaborate on two measurement techniques. In Section IV, we describe our prototype. In Section V, we present initial experimental measurements that demonstrate the feasibility of mid-call redirection. We conclude the paper in Section VI.

## II. THE ORBIT ARCHITECTURE

Our overlay solution consists of *intelligent endpoints* and *independent relays*. An endpoint is a host/application layer node that can act as the source or a destination of an IP Telephony call, and a relay is an application layer node with the capability to forward incoming packets to a specified destination. In our architecture, IP Telephony traffic either takes the direct IP path between the source and destination or goes through *at most* one relay. Each overlay path from the source to destination constitutes a *channel* between the source and destination. The direct IP path constitutes the direct channel and each relay provides an additional distinct channel. Relays in our architecture are independent entities and do not need to communicate or be aware of each other's presence.

Under normal network conditions, the direct channel is used. Since both endpoints passively monitor the quality of the current channel (based on the received packets), the receiver detects if the network QoS degrades (during an IP Telephony call) on the direct channel. A measurement phase is then initiated and, if a "better" channel is found, the stream is redirected via a relay. The forward and reverse channels may be through different relays, since each endpoint independently selects the channel to send its stream according to channel performance.

Only the endpoints need to be aware of a *limited set* of the relays. They pick the "best" channel to go around network QoS problems. Our architecture accommodates relays with differing capabilities, ranging from *dumb relays*, which only provide forwarding capability, to *cooperating relays*, which may participate in the channel measurements and/or keep some minimal state. Below we elaborate on the functional components of our architecture.

### A. On-Demand Measurement

In ORBIT, endpoints need to measure the quality of alternate channels only when the current channel's quality is unsatisfactory. Otherwise, no probe traffic is injected and the network behaves as though ORBIT were not deployed.

The criterion to determine when to initiate a channel switch can be done in several ways. For example, the receiving endpoint could continually evaluate the channel quality using the incoming RTP packets (e.g., to compute an  $R$  factor-like score [9]). When the quality (score) falls below a threshold, the overlay can be activated. This threshold can be pre-configured, determined at runtime and communicated to the endpoint periodically, or done at call setup time. It can be computed for the whole network or on a per-call basis. Alternatively, the initiation can be done by the end user explicitly. Once the

receiver decides that a switch is needed, it can either initiate measurements on the sender's behalf, or send a message to the sender that the quality degradation requires action.<sup>1</sup>

The actual measurement itself can be done in several possible ways; a couple of illustrative strategies are presented below in Section II-B and elaborated later in Section III. The switching decision is always executed by the sending endpoint.

We would like to point out one caveat to the rule of on-demand measurements. If the endpoints are sending traffic through a relay and there is a preference (e.g., due to policy) to switch back to the direct channel (when its quality improves), then the direct channel may be monitored by the endpoint even though the current channel is adequate with respect to quality. This probing can be done at a slower pace since the switch back to the direct channel is not critical given that the call is experiencing adequate quality.

### B. Channel Evaluation and Selection

Broadly speaking, the channel evaluation and selection techniques can use unique characteristics of the relays. Here we just present two possible measurement schemes.

In an end-to-end measurement technique that is suitable for use with dumb relays, a stream of packets are sent from sender to receiver via a relay. The packets are either echoed back to the sender (for evaluation), or the channel evaluation is done at the receiver and communicated to the sender, who then chooses the best channel. In another measurement scheme that is suitable for use with cooperating relays, the sender subscribes to a relay that can more efficiently probe endpoint subnetworks and communicate information about the likely channel state. These schemes are elaborated in Section III.

Note that an end-to-end measurement of a channel is, at a high level, reminiscent of endpoint-initiated admission control, and all prior work in that area (e.g., [10], [11] and references therein) can be effectively applied to this topic.

### C. Relay Forwarding Functionality

In our architecture, the decision to redirect IP Telephony traffic resides with the endpoints. However, the relay nodes need to somehow know the final destination of the IP Telephony packets they receive. The endpoints may convey this information to the relays in a number of ways. One method is to have stateless relays and tunnel packets through the relay using such schemes as IP-in-IP (or its variants). The tunnel is used only for redirection and any application-layer security is left untouched. Alternatively, relays can maintain state and exchange control messages with the endpoints. In particular, the relays can be triggered to provide forwarding functionality by using standard signaling protocol messages (e.g., H.323 or SIP). We provide more details on this subject in Section IV. Note that since our relays are independent, there is no need to maintain per-relay routing information.

<sup>1</sup>This control message can be sent more reliably via duplication through several relays. Also, the packet may be signed/encrypted and the session keys can be used for this purpose; another benefit of endpoint-initiated control.

## D. Overlay Management

In ORBIT, the relays are independent but the endpoints need to be told about the relays existing in the network and their various characteristics (such as IP addresses). This can be achieved in many ways, including receiving parameters through DHCP, being told by the IP Telephony system as part of the initialization mechanism, hardcoding an initial set of relays in the endpoints, etc. The relays may also be communicated during call setup time, in which case a network management system could provide the best possible set of relays for the specific call to the endpoint. In order to reduce overhead associated with communicating a large set of relay information to the endpoints, we can limit the number of relays to a small number. Our relay-placement studies indicate that a relatively small set of relays protects against single faults [7].

## E. Deployment

Intelligent endpoints can be deployed via software upgrades. The relay deployment depends on whether dumb or cooperating relays are required. For dumb relays, activating a tunneling capability such as IP-in-IP may be sufficient. For cooperating relays, the measurement support and forwarding functionalities can be put in different network components. The measurement functionality of relays, if needed, can be embedded via software in any device on the same subnet as the forwarder, potentially even in an IP Telephony endpoint.<sup>2</sup> IP Telephony gateways can, via software upgrades, be enhanced to perform relay measurement and/or forwarding capabilities via either stateless or stateful methods.

ORBIT allows incremental deployment of endpoints and the relays. For instance, it is possible to start with a few upgraded endpoints and a few relays in the network. Gradually new upgraded phones and relays can be added with little management effort. Furthermore, in most cases few or no new network components need to be added, so network management and operations costs for the enterprise are contained.

## III. CHANNEL MEASUREMENTS

In our framework, during an IP Telephony call, endpoints are aware of alternate overlay channels to the destination. When call quality on the current channel falls below a pre-defined threshold, the measurement phase begins to select a channel better than the current one. Below we describe two techniques to measure the QoS of a channel.

Both of the techniques described below utilize round-trip measurements when evaluating alternate channels. With some modifications, some of these strategies can be converted to measure one-way performance. We omit details for brevity.

### A. An End-to-End Measurement Technique, $E$

This measurement technique relies purely on endpoints and can work with dumb relays that simply forward probe packets. When the channel evaluation process starts, the source sends

<sup>2</sup>The measurement results in this case will not take the relay state into account, but will be a good approximation of the network path status.

probes to the destination through a set of channels (relays). The destination upon receiving the probes echoes them back to the sender. The channel information is encoded in the probe packet. The sender computes estimates of the round-trip delay, loss, and jitter using the probe responses.

In this scheme, the current channel need not be probed since the RTP stream serves as a “free” probing stream. When comparing the current channel to alternate channels, adjustments may be necessary since the probes are likely to be less frequent than the RTP packets. To compute comparable metrics across all channels, one option is to sample the RTP stream at the same rate as the probe frequency.

### B. Measurements with Summarization, $S$

This measurement scheme relies on the participation of relays as well as endpoints. The motivation behind *cooperating relays* is to reduce the probing overhead by summarizing statistics at the relays. The relays aggregate measurements at the subnet level and probe each subnet only once in order to assess network conditions between a given relay and endpoints on a given subnet. This approach reduces the probing overhead when many phones on a given subnet are simultaneously active and experiencing adverse network conditions.

The summarization mechanism follows a subscription-based model where endpoints contact relays to receive regular QoS updates pertaining to destination subnets. Endpoints send their subnet mask as part of the subscription for subnet identification. The relays maintain soft state about the subscribers and their subscriptions. A subnet is probed only when there is a subscriber who needs measurements to that subnet.

We describe the measurement protocol below. Let  $A$  and  $B$  be the source and the destination of an IP Telephony call.  $A$  is aware of three relays:  $R_1, R_2, R_3$ .<sup>3</sup>

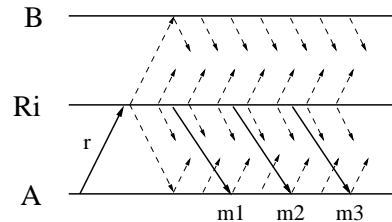


Fig. 1. Measurement Protocol

Figure 1 shows the messages exchanged between  $A$ ,  $B$ , and a relay  $R_i$ . Message  $r$  is  $A$ 's subscription request. Among other protocol specific information,  $r$  contains information on how often  $A$  would like to receive QoS updates. Let  $t$  milliseconds be the time interval at which  $A$  asks to be updated. Each  $R_i$ , upon receiving the request from  $A$  first checks if the subnets of  $A$  and  $B$  are being probed. If so, then  $R_i$  does not need to probe the endpoint whose subnet is already being probed. Otherwise,  $R_i$  starts probing the endpoints. In this example, assume that both  $A$  and  $B$  need

<sup>3</sup>The number of relays an endpoint is aware of is an adjustable parameter. Use of three relays here is for example purposes only. Note that the set of relays known to  $A$  and  $B$  do not have to be the same.

to be probed. Relay  $R_i$  starts sending probes to  $A$  and  $B$  at regular intervals. These probes are indicated with dashed lines originating from  $R_i$  in Figure 1. When  $A$  and  $B$  receive probes from a relay, they respond back to the sending relay. These messages are shown as dashed lines originating from the endpoints in Figure 1. Relay  $R_i$  computes raw performance metrics based on the response messages from  $A$  and  $B$  as they come in. Every  $t$  ms, each  $R_i$  computes estimates of round-trip delay, loss, and jitter between  $A$  and  $B$  via itself for the last  $t$  ms. These values are then reported to  $A$  every  $t$  ms shown as  $m1, m2, m3$ , etc. in Figure 1. Note that the protocol is symmetric and the steps for  $B$  are omitted here for brevity.

As with the previous measurement scheme ( $E$ ), the current channel is not probed.

### C. Comparison of Measurement Strategies

In this section, we compare the two strategies - end-to-end,  $E$ , and relay summarization,  $S$ . Strategy  $E$  has the benefit of not requiring any book-keeping at the relay and has advantages in some scenarios. However, strategy  $S$  benefits with lower probing cost, as analyzed below.

In this analysis, we consider the case where endpoints using the direct channel experience network problems and enter the measurement phase. We assume that  $n$  IP endpoints are involved in calls experiencing QoS problems in a given network at any point in time. We further assume that the endpoints of these calls reside in  $s$  subnets. For analysis, we assume that each endpoint is aware of all  $k$  relays in the network. We consider one QoS update interval corresponding to  $p$  probes. (Note that in our architecture, each endpoint can use a different set of relays, and  $p$  can vary by channel.)

We assume that any message between an endpoint and a relay has unit cost. (This can correspond roughly to the effective bandwidth consumed by the packet.) We assume that the cost of sending an end-to-end probe through a relay has cost  $c$  where  $c \geq 1$ .

For strategy  $E$ , monitoring an overlay channel requires forwarding each probe to the destination via a relay and receiving a reply at a cost of  $c$  each. The total cost per endpoint per QoS update interval through  $k$  relays is  $2kpc$ . The system wide cost is, therefore,  $Cost(E) = 2nkpc$ .

With Strategy  $S$ , each of the  $s$  subnets are monitored by each of the relays at a cost of  $2ps$  per relay for a total cost of  $2kps$ . With on-demand updates, endpoints request and are provided measurement updates at a cost of  $2kn$  per update period. System wide total cost is  $Cost(S) = 2kps + 2kn$ .

We define the improvement factor of summarization,  $R$ , as:

$$R = \frac{Cost(S)}{Cost(E)}$$

The value of  $R$  should be less than 1 for Strategy  $S$  to have a benefit over Strategy  $E$ . Substituting the cost figures from above we get:

$$R = \frac{2kps + 2kn}{2nkpc}$$

Since  $c \geq 1$ , breaking the above expression into 2 terms and simplifying we get:

$$R \leq \frac{s}{n} + \frac{1}{p}$$

From this analysis we conclude, as expected, that the benefit of Strategy  $S$  improves ( $R$  decreases) with increasing the number of calls per subnet and increasing the probing frequency. In [8], we present some simulation results that show that these proposed measurement techniques can detect network impairments rapidly and rescue IP Telephony calls in sub-second intervals.

## IV. PROTOTYPE IMPLEMENTATION

We implemented a prototype of our system. The endpoint functionality of our system is implemented as a *bump*, i.e., a bump-in-the-wire that can be placed in front of any device, and in particular, any commercial off-the-shelf real-time IP device like an IP phone. This is possible since our overlay system does not affect the content of the packets but only their paths.

The bump is implemented as an application under Linux and functions as a device-specific bridge. It uses Linux's raw sockets to capture packets promiscuously, to bridge packets, and to generate packets as required for channel measurement. The bump has logic to automatically learn the MAC address, IP address, netmask and Layer-2 options (e.g., 802.1Q settings) of both the real-time device it is connected to and the upstream router. It detects RTP streams using a simple statistical analysis of bridged packets that takes into account the size and inter-arrival time of packets. For deployment ease, we run our bump on small two-interface single-board computers [12], [13]. The small form factor of the bump and our implemented features ensure that the bump can be placed in front of any device, e.g., an IP phone or computer, (without the need for *any* configuration) and achieve the endpoint functionality of our system for that device.

Conceptually, the channel evaluation and redirection features of a relay could be separated. However, in our prototype, we use the same device for both functions. The channel evaluation feature of the relay is implemented as a Linux application. The application supports measurement summarization as described in Section III-B. To avoid maintaining a list of subscribing endpoints at the relay, we implemented the scheme where the relay sends measurements to an endpoint only when requested. In our prototype and experiments, we use IP-in-IP for redirecting packets through a relay channel. We use the standard Linux `ipip` module to implement the redirection function on the relay.

In our prototype, during channel evaluation, the channels are assigned scores as follows. Let the "probable-loss" parameter  $\ell$  be the time since the last probe was received *if this interval is greater than two probe periods*; otherwise,  $\ell$  is set equal to 1. Also, let  $d$  be the delay on the channel path. Each channel  $i$  is assigned a score,  $Score_i = cost \cdot d \cdot \ell^3$ , where  $cost$  is the administrator-specified cost of the path. This simple function weights the consecutive loss heavily (as a cube) and the delay

linearly. The function was chosen intuitively for the prototype and takes into account the significant impact of loss on IP Telephony quality; we are currently exploring several other channel evaluation functions.

## V. EXPERIMENTAL ANALYSIS

We analyzed the impact of our overlay solution via experiments on a testbed consisting of two commercial SIP phones connected by a network of three routers as shown in Figure 2. The phones had a bump in front of them and were connected to routers R2 and R3, respectively. A relay and a SIP proxy were connected to R1. The preferred path for the phones to send RTP streams to each other was through the link between R2 and R3. The path through R1 (and the relay) provided the redundant path. We ran `nistnet` [14] on the routers and could emulate a network impairment of delay, loss and jitter. This also implied that the links could emulate WAN characteristics and the testbed represented a three-site network connected by WAN links.

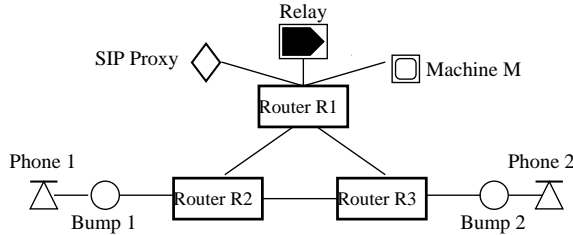


Fig. 2. Experimental testbed.

The system rapidly and consistently switched over to the channel through the relay when an impairment was introduced on the link between R2 and R3. A listener at the receiver phone would only notice a slight glitch in the audio stream.

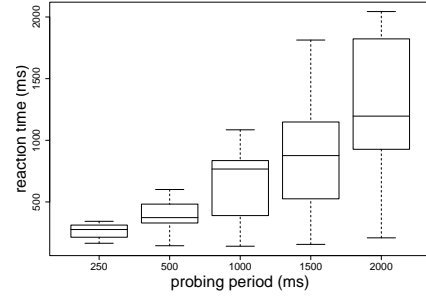
The receiver detected the network problem by monitoring the incoming RTP packets. The high packet rate of the stream (20 ms.) allowed problem detection time in the sub-second range. For example, with an  $R$ -factor threshold of 80, we observed detection times less than 200ms. Since detection time is independent of measurement strategies, in the experiments we focus on the reaction time for brevity.

To quantitatively measure the impact of various factors on the solution, we varied several parameters and computed the recovery time. This was done by having a machine  $M$  connected to R1 send an impairment packet to a server on R2 (to which the bump of the phone acting as the sender was connected). The router sent a confirmation as soon as it introduced the impairment. When the bump decided to switch, it sent a UDP packet to machine  $M$ . The difference between the reception times of the two packets (impairment confirmation and switch message packets) estimated the reaction time. Note that both packets (the impairment confirmation and the message from the bump) travel the same path, and the estimate of reaction time we make is therefore quite accurate. Our experiments and results are reported below. For the experiments reported here, the probing frequency is

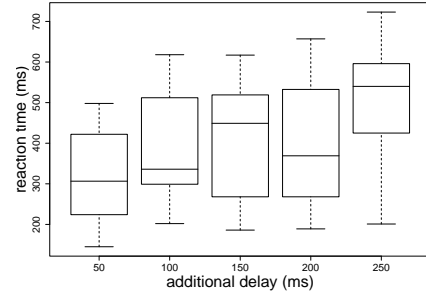
the same as the update frequency (see Section III-B). Each experiment was repeated 100 times.

### A. Probing Frequency and Added Delay

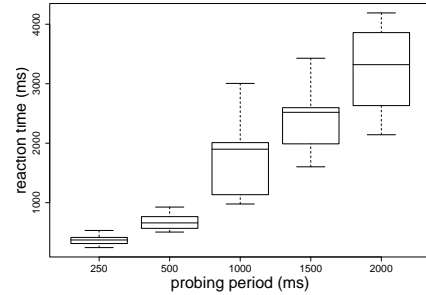
We introduced an additional delay of 100 ms on the direct path from source to destination and measured the response time for various probing periods.



(a) Added delay of 100 ms.



(b) Probe frequency of 500 ms.



(c) Network fault (100% loss)

Fig. 3. Variation of Reaction Time. In each box plot, the whiskers extend from the 1<sup>st</sup> to 99<sup>th</sup> percentiles, the box extends from the 25<sup>th</sup> to 75<sup>th</sup>, and the median is marked with a line.

We observe that the response time increases almost linearly with increasing probe intervals, as one would expect based on our channel evaluation formula from Section IV.

### B. Network Delay

We kept the probe frequency constant at 500ms and measured the reaction time while varying the added delay. We observe that the reaction time increases with delay. This is because the decision to switch is delayed by the amount by which the probe is delayed.

## C. Network Faults

We introduced a network fault (100% loss) on the direct path from source to destination and measured the reaction time while varying the probe frequency. We observe that the reaction time increases linearly with increasing probing period. Our channel evaluation function is designed to rapidly react to consecutive packet losses. Such losses are detected when consecutive probes fail to receive responses over a number of probing periods. Hence, it is expected that the reaction time is a function of probing period.

Our experiments show that the response of our system to network impairments is very fast, demonstrating the feasibility of using ORBIT for mid-call recovery for IP Telephony.

## VI. CONCLUSION

In this paper, we presented a simple, practical on-demand overlay solution for real-time traffic (specifically, IP Telephony) in enterprises. An application-layer solution provides tremendous benefits, since the application can best decide what quality it receives and expects. By having a simplified architecture involving intelligent endpoints and independent relays, we derive the benefits in terms of optimized techniques and streamlined implementation.

We have implemented our system and have presented initial experimental results based on our prototype implementation. The experiments suggest that IP Telephony is indeed protectable with traffic redirection that can take place in sub-second to seconds range.

In the future, we plan to further experiment with our system, study different algorithms for channel selection and evaluate them. We also plan to implement our endpoint functionality inside the IP phone and thereby use the state of the phone's protocol stack to make more intelligent decisions.

## REFERENCES

- [1] A. A. P. Pan, G. Swallow, "Fast Reroute Extensions to RSVP-TE for LSP Tunnels," in *IETF RFC 4090*, 2005.
- [2] *IEEE Journal on Selected Areas in Communications*, Jan 2004.
- [3] S. Tao, K. Xu, A. Estepa, T. Fei, L. Gao, R. Guerin, J. Kurose, D. Towsley, and Z.-L. Zhang, "Improving VoIP Quality Through Path Switching," in *Proc. INFOCOM*, 2005.
- [4] S. Savage, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vahdat, G. Voelker, and J. Zahorjan, "Detour: Informed Internet Routing and Transport," *IEEE Micro*, vol. 10, no. 1, pp. 50–59, 1999.
- [5] D. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris, "Resilient Overlay Networks," in *Proc. 18th ACM Symposium on Operating Systems Principles*, Banff, Canada, October 2001.
- [6] J. Han, D. Watson, and F. Jahanian, "Topology Aware Overlay Networks," in *Proc. INFOCOM*, 2005.
- [7] B. Karacali, M. J. Karol, P. Krishnan, K. Kumar, and J. Meloche, "Independent Relays in Overlays for Reliable Enterprise IP Telephony," in *Proceedings of the 2006 Sarnoff Symposium*, March 2006.
- [8] B. Karacali, M. Karol, P. Krishnan, K. Kumar, and J. Meloche, "Measurement Techniques in On-Demand Overlays for Reliable Enterprise IP Telephony," in *Proceedings of the IEEE International Conference on Communications (ICC)*, 2006.
- [9] ITU-T Recommendation G.107, "The E Model: A computational model for use in transmission planning," 2003.
- [10] L. Breslau, E. W. Knightly, S. Shenker, I. Stoica, and H. Zhang, "Endpoint admission control: Architectural issues and performance," in *Proc. of SIGCOMM*, 2000, pp. 57–69.
- [11] K. Mase and Y. Toyama, "End-to-end measurement based Admission Control for VoIP networks," in *Proc. of ICC*, 2002, pp. 1194–1198.
- [12] Soekris Engineering. [Online]. Available: <http://www.soekris.com>
- [13] Axis Engineering. [Online]. Available: <http://www.axis.com>
- [14] National Institute of Standards and Technology . [Online]. Available: <http://www-x.antd.nist.gov/nistnet/>