

Fault-Tolerant Overlay Protocol Network

Nicholas J. Shelly, Nathan A. Jensen, Leemon C. Baird, and Jason A. Moore

Abstract—Voice over Internet Protocol (VoIP) and other time critical communications require a level of availability much higher than the typical transport network supporting traditional data communications. These critical command and control channels must continue to operate and remain available in the presence of an attack or other network disruption. Even disruptions of short duration can severely damage, degrade, or drop a VoIP connection. Routing protocols in use today can dynamically adjust for a changing network topology. However, they generally cannot converge quickly enough to continue an existing voice connection. As packet switching technologies continue to erode traditional circuit switching applications, some methodology or protocol must be developed that can support these traditional requirements over a packet-based infrastructure. We propose the use of a modified overlay tunneling network and associated routing protocols called the *Fault Tolerant Overlay Protocol* (FTOP) network. This network is entirely logical; the supporting routing protocol may be greatly simplified due to the overlays's ability to appear fully connected. Therefore, ensuring confidentiality and availability are much simpler using traditional cryptographic isolation and VPN technologies. Empirical results show for substrate networks, convergence time may be as high as six to ten minutes. However, the FTOP overlay network has been shown to converge in a fraction of a second, yielding an observed two order of magnitude convergence time improvement. This unique ability enhances availability of critical network services allowing operation in the face of substrate network disruption caused by malicious attack or other failure

Keywords—Availability, Overlay, Tunnel, Voice over Internet Protocol, Virtual Private Networking

I. INTRODUCTION

In its cover article in September 2005, the *Economist* reports “how the Internet killed the phone business” in discussing the unassailable advance of VoIP in the field of voice communication. Many in the phone industry predict that by 2010 telephony will be offered alongside web browsing as a free service as an incentive to purchase broadband access or pay-per-view services[5]. Nonetheless, for VoIP to assume mission critical communications, such as emergency services and military communication between commanders, (thus making traditional circuits obsolete), the reliability must be drastically improved. The coveted “five nines”, or 99.999% availability of conventional circuit-switched communication seems an ambitious benchmark in today’s Internet in which frequent updates, link faults, configuration errors, and malicious attacks can easily discontinue a voice conversation. However, the newly proposed and developed FTOP network alleviates such reliability is-

ues through split-second convergence by employing a systematic check of the direct logical path to the destination and indirect logical paths to all other nodes in the overlay network.

The three primary tasks of the FTOP are to detect a connection failure in the direct logical source-destination path, discover valid and available intermediate nodes, and force packets through the intermediate node paths until the direct path returns to service. Our goal is to make fault detection and re-routing fast enough to sustain a voice or other time critical connection. To meet the first and second requirements we execute a user-level program to send path status information to all other nodes to create an adjacency matrix used in the new system-level tunneling protocol. This user level Connection Checker requires minimal bandwidth as each node sends out its current status at a controlled frequency, balancing bandwidth with a faster convergence time, which the receiving node will simultaneously use to recognize its availability and its current paths (see Section III-C). If there is excessive latency, power outages or any other obstacle to smooth communication, the checker will label the path as “down” (a ‘0’ in the matrix). In accomplishing the third task, the kernel level FTOP algorithm uses the path adjacency matrix, updated through a series of system calls by the user-level Connection Checker, to determine where to send the packet. FTOP looks up the network and confirms its availability before deciding on the path, with negligible change in total travel time. Results have shown in a small scale network an improvement in convergence time by a factor of over 750. Through the use of only one intermediate node or FTOP router, packets could be sufficiently forwarded in the case of path failure or communication latency. Through broad geographic diversity of the FTOP routers, the effect and resilience of the FTOP network will be magnified making the “five nines” requirement by emergency services, military commands and other essential communication possible with VoIP.

II. BACKGROUND AND PREVIOUS WORK

A substantial amount of work has already been accomplished in regard to overlay networks and traditional quality of service implementations. The following sections contain a brief discussion on previous overlay network efforts and traditional quality of service methodologies. To properly frame the benefits of the FTOP network, and the dif-

ferences between the FTOP network and other attempts to address reliable communications on an inherently unreliable network, it is useful to examine other efforts in the area.

A. Motivation for Work

The Department of Defense has a great interest in using VoIP to support both mission critical command and control communications as well as support and managerial traffic. The end goal is a “converged” transport network that supports many services, (voice, video, data, etc), on the same infrastructure. To that end, the Joint Test Interoperability Command (JTIC) has developed guidelines enumerating the requirements that a Command and Control Voice Grade Local Area Network (C2VGLAN) must meet. These requirements were added as Appendix 3 of the General Switching Requirements Document (GSCR). One of these requirements is a two second convergence time for the applied routing protocol[4]. Several technical solutions have been developed that can meet these criteria , but these implementations differ significantly from normal network configurations found in use today[8]. Typically, current networks do not have the redundancy required of a C2VGLAN; furthermore, all of these solutions are strictly approved for use on a LAN that connects to traditional circuit switching infrastructure for long haul connectivity. Approved VoIP packages are only being used “inside the gate,” or very limited geographical areas. These inside the gate solutions may not be easily integrated into existing networks. Most available approved C2VGLAN solutions have very few layer three hops; they are very reliant on layer two and use of Virtual LANs (VLANs) to segment traffic. It is very difficult, if not completely impossible, to keep outages under two seconds if the core network infrastructure is several layer three hops removed from the end user devices.

B. Overlay Network Efforts

Overlay networks are becoming more common every day. Virtual Private Networks (VPNs) are now the standard for remote access rather than an esoteric magic trick for the technically adept[7], [6], [1]. Using Resilient Overlay Networks (RONs) to overcome weaknesses of the substrate transport network has also been attempted in the past with some promising results. It has been shown that tunnel networks can route around network outages that lead to extended outages on the internet mainly due to BGP convergence time. However, these efforts have not been centered on time critical protocols; the average time to detect an outage on a RON network is stated to be 19 seconds[2], (too long to maintain a voice call). One critical finding did come from this work: most network outages can be overcome by injecting one alternate hop into the route. If there is at least one intermediate node reachable by the

source and the destination, using this single intermediate² node will provide source to destination connectivity[2].

C. Traditional Quality of Service

Recently, much emphasis has been placed on producing Quality of Service (QoS) implementations on inherently unreliable IP networks. However, nearly every attempt to implement QoS on IP has been focused on some variant or combination of priority queueing, resource reservation, and controlled link sharing[3]. While very useful, these priority queueing implementations do not, nor can they, ensure that a time critical connection can remain intact while a routing protocol converges after a network failure.

One interesting future effort could include a traditional priority queue QoS implementation running on an FTOP network. A packet’s quality of service tags are generally not evaluated when it is routed through the internet (every customer would request the highest priority). However, using an overlay network, it is possible to preserve the quality of service settings through the internet. These quality of service settings do not change the way the internet handles the priority of the packet, but it may be useful once it reaches another overlay node or network. If a service level agreement can be reached by the customer and service provider that includes assurances of bandwidth along specified routes and a willingness to participate in QoS methodologies, this area of research becomes more promising[9].

III. THE FAULT TOLERANT OVERLAY PROTOCOL NETWORK

The FTOP network concept relies on simplifying assumptions made about a fully connected network. In the following sections, first we cover some general routing concepts, followed by a brief discussion of these simplifying assumptions made on the logical FTOP network. Since the FTOP implementation requires significant modification of the kernel and network stack, a small description of the network stack and data flow is also provided. Following this information, the implementation of the FTOP in both user and kernel space is given.

A. Routing Theory

A dynamic routing protocol is a method for routers on a network to discover other remote networks by advertising networks they can reach to neighbor routers. The neighbor routers propagate these advertisements allowing all routers to reach all remote networks. In general, this is a mature field, and very robust implementations are in use on today’s networks, (RIP, EIGRP, OSPF, BGP, etc). The time taken for all routers to propagate advertisements and gain an accurate picture of the network is called convergence time. All of these protocols have one thing in common: they must support routes from source to destination of ar-

bitrary length. Hence each of these algorithms may take a significant amount of time to converge as routes are passed along these paths.

An overlay or tunnel network can be leveraged to mitigate the weaknesses inherent with these substrate networks and the routing protocols that support them. Tunneling is the ability to package one packet within another packet, route it through the substrate network, un-package it when it reaches its tunnel destination, and then route it to its final destination. Using this methodology it is possible for nodes that are separated by many hops on the substrate network to appear as though they are directly connected. Many of the nodes can be joined to form an overlay tunnel network where hop count between nodes is completely unrelated to the hop count on the underlying substrate network. The critical piece of information here is that nodes can be in physically diverse remote locations and yet appear to be directly connected to any other overlay node. See Figure 1.

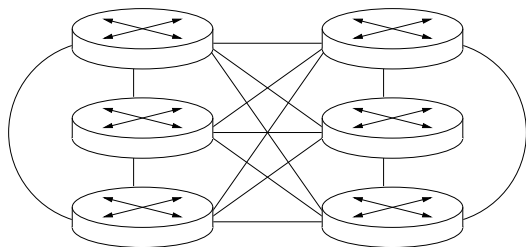


Fig. 1. Fully Connected Six Node Network

Given a network where any node can directly reach any other node, some simplifications can be made to a routing protocol to greatly reduce or nearly eliminate the time it takes to converge to an accurate reflection of reality. Given a particular node, every other node on the overlay is in one of two states: reachable or unreachable. This attribute alone is not sufficient to build a fast converging fault tolerant network. If routers simply try to forward all information directly to the destination, a failure on the substrate network can and generally does result in a failure on the overlay network. Once the substrate network routing tables converge, traffic will again flow from source to destination through the directly connected tunnel nodes. On the other end of the spectrum, a full-blown arbitrary path length routing protocol can be used on the overlay network. Again, little is gained here as convergence on the overlay network itself can be too slow to support a time critical connection.

The desired outcome is a protocol that can force traffic to take advantage of physical diversity on the substrate network, yet still be lightweight and converge very rapidly. Since overlay hop count is not related to substrate hop count in any way, forcing traffic through one intermediate node can inject a huge amount of physical diversity into the

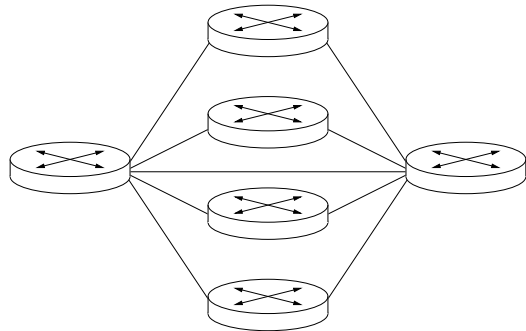


Fig. 2. Logical Overlay Routes from Source to Destination

utilized path on the substrate network. Therefore, it is hypothesized that a routing protocol that only tracks paths of length two on the overlay network can both converge very quickly and take advantage of large amounts of physical diversity on the substrate network. Given any source and any destination, the routes that may be utilized are shown in Figure 2. The routing algorithm becomes greatly simplified with little overhead and still utilizes the diverse routes on the substrate network. As described below, there are very few routing decisions to make when forwarding a packet:

- Did the packet originate from a subnet local to this router?
 - Yes: This is the first router encountered. Is the direct route to the final destination up?
 - * Yes: Send the packet directly to destination
 - * No: Send the packet to an intermediate node that can reach the final destination.
 - No: This router is an intermediate hop. It must forward the packet directly to the final destination.

Using FTOP to route voice traffic overcomes some of the limitations observed in current inside the gate solutions. Not only does it allow for a much more complicated layer three network, it is one step along the way to routing voice communications through Internet WAN links. Empirical results suggest that the FTOP network routes will converge in less than a second provided that there is a logical path of length two or less through the network. More specifically:

- If:
 - $\forall (S, D) \exists I \text{ s.t. } l(S, I) = TRUE \text{ and } l(I, D) = TRUE$
- Where:
 - Source Node: S
 - Destination Node: D
 - Intermediate Node: I
 - Physical Path Check Function: $l(Node_0, Node_1)$
 - * $l(Node_0, Node_1) = TRUE$ when path is connected
 - * $l(Node_0, Node_1) = FALSE$ when path is down
- Then:
 - A logical path of length two or less exists
 - Network will converge in less than one second

If one considers this property of the FTOP network when designing physical substrate LANs and the placement of intermediate nodes, it is possible to generate a very robust voice communications capability on top of a less reliable substrate network. If physically diverse routes exist on the substrate network connecting the source, the destination, and at least one common intermediate node, a link failure cannot result in a voice outage longer than two seconds.

B. The FreeBSD Network Stack

Before one can discuss our particular implementation of the FTOP, it is useful to conduct a brief explanation of the inner workings of the FreeBSD stack. The construction of the FreeBSD stack is interesting in its own right; it is a layered collection of functions that move data from application to physical layer and vice versa. The implementation of each layer is abstracted away from adjacent layers, yet each layer retains function pointers to appropriate functions in those adjacent layers to move data up and down the stack. This layered approach makes further modification and development of new protocols significantly less challenging. A chart depicting information flow through the BSD stack is contained in figure 3.

C. The Connection Checker

The connection checker relies on the ability to find, track, and update paths of length two on the overlay network very quickly. Therefore, a connection checking program that runs in user space was developed. For an n node network, this connection checking program maintains an $n \times n$ two dimensional array containing the status of paths between nodes. At a defined interval, this array is sent to every other participant on the overlay network. In the proof of concept implementation, each element in the array is a four byte integer. Clearly, a more efficient data structure could be used for this transmission. However, as discussed in the analysis section, this implementation scales well even with this inefficiency remaining from the beta code. The cost, (shown in equation 1), of this path checking implementation depends on the send frequency, number of overlay nodes, and header sizes.

$$Bps = f_{send} (n - 1) (4n^2 + IP_{hdr} + UDP_{hdr}) \quad (1)$$

The receive side of the connection checker is slightly more complicated. This segment of code continually looks for incoming messages from the other overlay nodes. When it does receive a message, it notes the time of reception and updates the local two dimensional array to indicate a successful receive. If the checker does not receive a message from a particular node within a specified time, the checker updates its local two dimensional array to indicated that it cannot receive from the remote node in question. It is assumed that any overlay member can send/receive to/from

themselves, (the diagonal of the two dimensional array)⁴. For a path to be considered up, each node must be able to receive from the other. For example, in Figure 4 the path between node two and node four is up, while the path between node three and node four remains down because at least one node cannot receive.

$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

Fig. 4. Example Path Array Matrix

At user adjustable specified intervals, this user level application makes kernel system calls. This system call passes the node's current path array matrix into kernel space. The new kernel module uses this array to make routing decisions as discussed in the routing theory section.

D. The Kernel Module

One desirable trait of any new routing protocol is that it be completely transparent to the edge devices and user applications. From the user's perspective it should behave no differently than the traditional network stack; this allows any off-the-shelf application to take full advantage of the fast converging overlay without modification. Therefore, this new protocol requires the creation of a new kernel module. This module is loosely based on the FreeBSD Generic Routing Encapsulation (GRE) module that comes standard with most recent FreeBSD releases.

The final fast converging overlay product shares some code with its parent, specifically portions that create new headers and other menial bookkeeping. It also falls into the network stack previously described in exactly the same manner as the original GRE package. However, the source and destination addresses for the original GRE package were determined when the tunnel was initialized. It had no ability to evaluate a packet that was to be tunneled and intelligently select an alternate tunnel endpoint. Furthermore, the GRE package did not have any system calls built into it. System calls had to be created to pass the two dimensional matrix from the user level path checking application into kernel space.

Upon kernel module insertion and initialization, overlay network participants are read in from a text file. This is essentially a list of substrate network IP addresses. Each member router may or may not have an overlay subnet attached to it. If the member router does own an overlay subnet, the entry in the text file will also contain a network address and subnet mask. In this manner, each kernel module can determine:

1. Which overlay networks/subnets are directly connected to this router.
2. To which substrate network tunnel endpoint should a

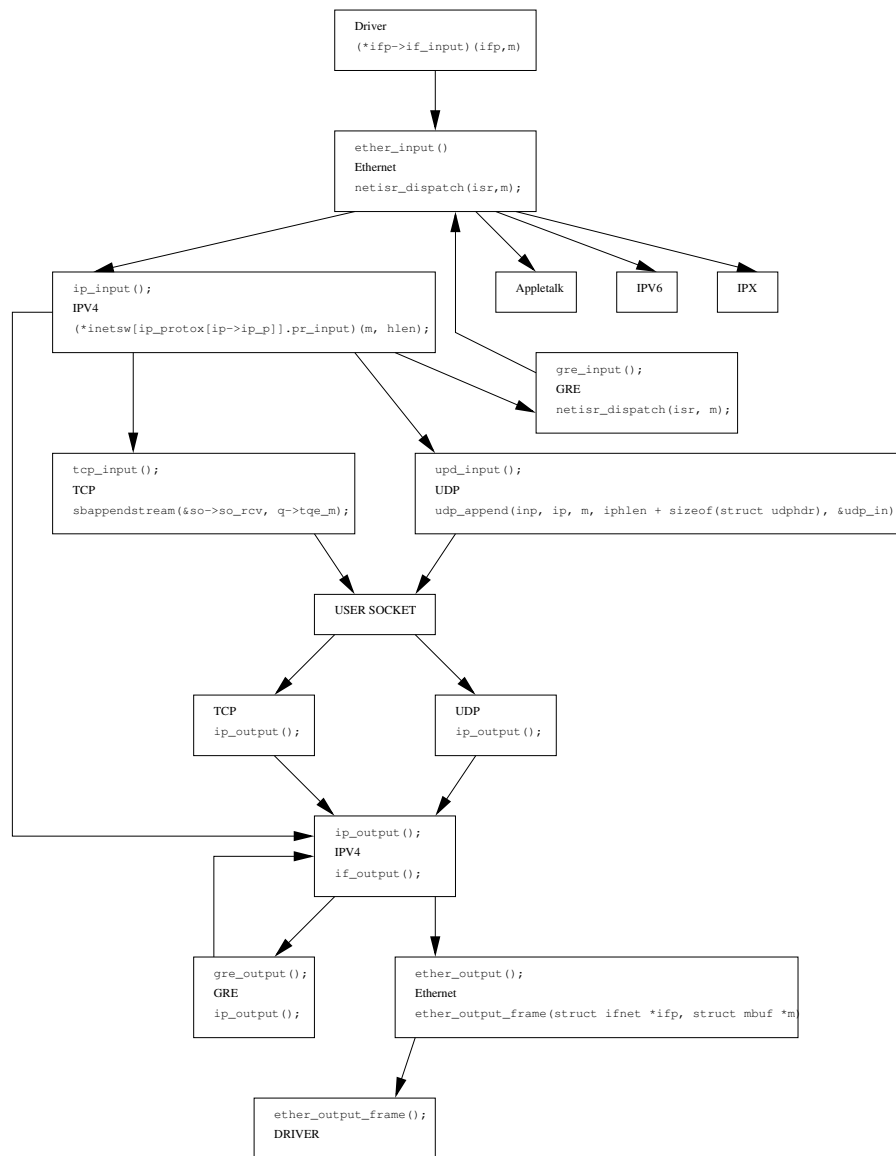


Fig. 3. FreeBSD Network Stack Flowchart

packet be sent to ensure that it arrives at the correct overlay network/subnet.

Upon receiving a packet, the new kernel module does traditional netmasking using the information from the text file to determine if it originated from a local subnet. If it did come from a local subnet, the router will check the two dimensional path array to see if the direct path is up. If it is, it will send the packet directly to the tunnel endpoint that owns the destination subnet. If not, it will search the matrix to find the first path of length two that is up. It will send the packet to the substrate network tunnel endpoint of the intermediate node. If there are no paths of length two available, the packet is discarded. If the packet did not originate from a local network/subnet, then the packet must be forwarded directly to the destination tunnel

endpoint. If that path has failed, (failure occurred after the original router sent the packet to the intermediate node in question), then the packet is lost.

IV. EMPIRICAL RESULTS

A. Convergence Times

The main metric of concern in this research effort is convergence time, (how long it takes to detect a link failure and choose an alternate path). Typical convergence times of a complex BGP network may be as high as six to ten minutes[2]. To test the convergence times of the FTOP network, a test perl script was developed that passed traffic from a typical windows client behind one FTOP node, through the substrate network (perhaps being routed

through an intermediate overlay node), to the correct tunnel endpoint, and finally to a client behind a different FTOP node. This script would send and receive packets as fast as the client could generate them. Errors were induced on the substrate network by physically disconnecting one or more of the links. After the network converged from an induced failure, the link was restored and the process was repeated. Figure 5 shows a histogram of the convergence times during 75 different network disruptions. Note that this is a worst case convergence time; it is truly a measure of convergence time in addition to the time that it took the perl script to send the next packet. The time shown is for complete communications restoration. At no time did convergence take any longer than .9 seconds, and on average it was approximately .6 seconds.

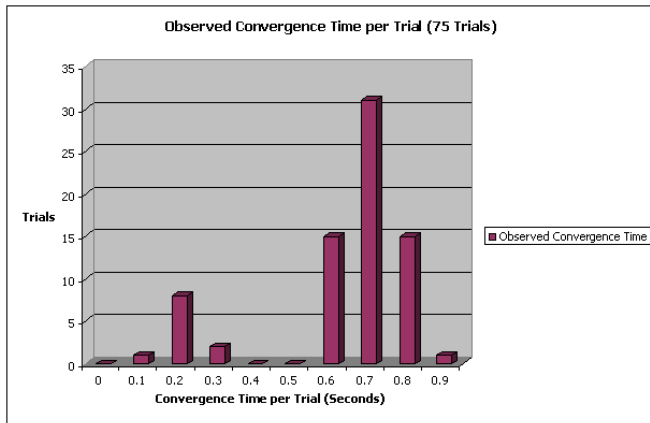


Fig. 5. Convergence Times for FTOP Network (Compared to typical times of 6-10 minutes for the underlying network)

B. Packet Loss and Latency

Packet loss is another interesting metric gathered from the FTOP network. The FTOP network lost an average of 4 packets per outage when the client was producing packets as fast as its software and hardware allowed. What is even more interesting is that packet loss was very bi-modal. It appeared that an outage would cause 1-2 packets to be lost, or it would cause 7-8 packets to be lost. Further investigations discovered that when the smaller amount of packets were lost, a link was broken between interfaces that were integrated into the motherboards of the routers. If a link was severed that included an interface that was connected to the router via a PCI slot, then a larger amount of packets were lost. This leads one to conclude that the amount of buffering done on the card, and the card's ability to notify the kernel that the interface was down is currently the limiting factor on convergence time and lost packets rather than the FTOP protocol in use. A histogram describing these results is shown in figure 6.

Latency is another critical element that must be evaluated when analyzing the FTOP protocol. Placing a rout-

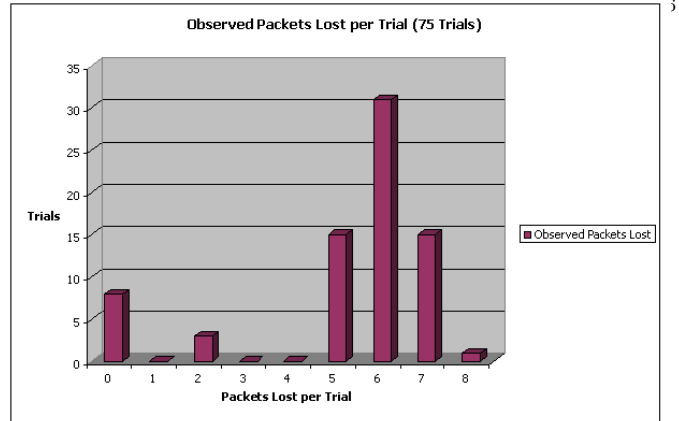


Fig. 6. Packets Lost During Convergence

ing decision in the FTOP package increases latency by one traversal of the FreeBSD network stack, (one internal layer three hop). The FTOP routing decision is a table lookup and can be done in linear time just as any typical layer 3 routing decision. An attempt was made to measure the additional overhead of this step. However, using the C time system calls lacked the clock granularity to show a statistically significant increase in time. For connections traversing the direct logical path there was no measurable difference between FTOP and non-FTOP latencies. When path indirection occurred, an indeterminate number of additional physical hops are included in the path. This can increase latency significantly. However, the reason the redirection occurred in the first place was due to a path failure in conjunction with a non-converged substrate network; a non-FTOP routing protocol would have dropped the connection entirely. Once the direct link comes up, latency will return to that of the direct path link.

V. CONCLUSION AND FUTURE WORKS

The FTOP routing protocol is a lightweight routing protocol. It consists of a loadable FreeBSD kernel module and a user space executable. It takes advantage of tunneling techniques to create a fully connected overlay network. Empirical evidence suggests that it can detect and route around network outages quickly enough to sustain a voice connection (generally two orders of magnitude quicker than traditional routing protocols). Follow-on work for this project is to remove the user level executable and use actual message traffic to indicate the existence or non-existence of path connectivity. Another future change is to take advantage of the many logical paths available. The data stream could be spread across these logical links for even greater reliability; received packets could be used to reconstruct lost ones effectively covering the small outage times observed while the FTOP protocol converges.

REFERENCES

- [1] D. Andersen, H. Balakrishnan, M. Kaashoek, and R. Morris. The case for resilient overlay networks, 2001.
- [2] David Andersen, Hari Balakrishnan, M. Frans Kaashoek, and Robert Morris. Resilient overlay networks. In *Proc. 18th ACM Symposium on Operating Systems Principles*, Banff, Canada, October 2001.
- [3] R. Braden, D. Clark, and S. Shenker. Integrated services in the Internet architecture: an overview. Technical Report 1633, 1994.
- [4] Joint Test Interoperability Command. General switching center requirements: Appendix 3, September 2003.
- [5] The Economist. How the internet killed the phone business, September 2005.
- [6] B. Gleeson, A. Lin, J. Heinanen, and G. Armitage. A framework for ip based virtual private networks, 1998.
- [7] Douglas Maughan, Mark Schertler, Mark Schneider, and Jeff Turner. Internet security association and key management protocol (ISAKMP). Internet Draft (draft-ietf-ipsec-isakmp-08), 1997.
- [8] Randall Perry and John Lindamood. Cisco ip communications and jtic certifications update, January 2006.
- [9] Jingdi Zeng and et al. Toward ip virtual private network quality of service: A service provider perspective.