Application Layer Striping: A Deployable Technique for Providing Quality of Service

Raheem Beyah, Raghupathy Sivakumar, and John Copeland
School of Electrical and Computer Engineering
Georgia Institute of Technology

Abstract - **In this paper we propose a deployable approach to improving QoS by using the familiar overlay architecture paradigm.** ***The goals of this work are to 1) create an overlay architecture which allows us to sample specific path quality metrics among different paths; and 2) utilize the proposed overlay architecture in order to implement our proposed QoS-based routing scheme, Application Layer Striping (ALST).*** **We show that we are able to achieve better than best-effort QoS without modifying intermediate nodes (i.e., routers), thus encouraging immediate deployment. Additionally, this research is performed on an actual wide area network testbed, comprised of universities across the nation. Also, we assemble this architecture as a peer-to-peer framework, encouraging collaborating individuals with average workstations to improve the QoS of their traffic.**

I. INTRODUCTION

The Internet has evolved to provide a multitude of services. In order to accommodate desired real-time services, or to merely achieve better than best-effort service[1], some level of Quality of Service (QoS) must be established. Several different techniques have been proposed to resolve this issue, all of which require a moderate to significant overhaul of the current Internet infrastructure. We propose a deployable approach to improving QoS by using a generic, extendable, overlay architecture; the Generalized Application Layer Overlay (GALO)

In this paper, we present findings from a multiple week study of Internet traffic dynamics using link throughput as the primary metric. Many traces were conducted through a four node, mesh-connected, "real world" Internet testbed. We generate different types of traffic using a generic, multipurpose, sockets program. Among other things, these findings show improper load balancing and link utilization across Internet paths. As motivation for our work, we synthetically construct alternate paths, showing that all of the nodes would have a better path quality if striping over alternate links were used, with a potential increase in throughput seen by a flow, ranging from 10% - 400%.

The aforementioned findings give initial motivation for the establishment of GALO. GALO provides a mechanism to obtain the state of the monitored network via invasive or non-invasive sampling/probing. The sampling/probing is performed by cooperating end nodes. The GALO architecture is designed in a modular fashion and has three primary components: Distributed Client Engine (DCE), QoS Routing Engine (QRE), and the Forwarding Engine (FE).

We extend GALO by introducing the *Application Layer Striping* module *(ALST)*. Application Layer Striping is a concept introduced to provide a better than traditional, shortest path, best-effort, routing service to traffic flows. ALST is considered a form of QoS-based routing that has the ability to utilize multiple alternate paths to provide an overall better path quality. It can make routing decisions based on more than one metric (i.e., shortest path, loss, delay, delay jitter, and throughput), as well as help to distribute the traffic load among multiple nodes and links in the Internet. This is achieved by constructing non-shortest path routes by appending different link combinations, with specific path quality, from the source node to the destination node. Using end nodes as switches, we then simultaneously stripe data over multiple alternate paths. We show empirically, that by simply striping data over alternate, underutilized links, we can, in a deployable fashion, immediately improve QoS. The performance analysis of ALST shows that 90% of the striped flows experienced a 10%-350% increase in throughput over the average throughput of the default path.

The remainder of this paper is organized as follows: Section II discusses traffic measurement, our experimental methodology, findings from the empirical traffic analysis conducted on our testbed, and the construction of synthetic alternate paths. In Section III, we discuss the motivation and description of the GALO architecture. Section IV discusses an extension to GALO, the ALST module. Section V gives the performance analysis using ALST, and Section VI concludes the paper.

II. ALTERNATE PATH EVALUATION AND MOTIVATION

A. Background

Normally, researchers perform empirical studies to understand precisely the behavior of the network of interest. These studies are generally helpful, but are most accurate for the particular region of the Internet at the particular time of measurement. Likewise, our study is most accurate for its focus region at the times of measurement. From such focused study, we can only glean and infer principles and characteristics of general Internet traffic behavior. Additionally, empirical studies must be performed regularly, due to the inability to predict future types of Internet traffic and their characteristics.

In our experiment, we use a generic sockets program to empirically observe Internet traffic dynamics, using

[1] Better than best-effort QoS will not suffice for real-time, delay sensitive, traffic; but can be beneficial for non real-time traffic, e.g., mail transfers between mail servers, or FTP flows.

0-7803-8199-8/03/$17.00 © 2003 IEEE

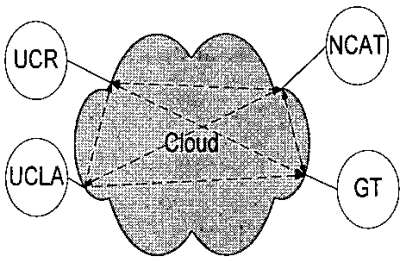

Fig. 1. QoS WAN testbed.

throughput as the primary metric in order to observe traffic load imbalance over our testbed.

B. *Experimental Setup*

We used four end nodes, mesh connected via the Internet, to conduct our experiments. As such, each node takes a different route to every other node. The experiments were bidirectional, so in total, twelve different "paths" were monitored. The nodes were stationed at universities (US) on the east and west coast: Georgia Institute of Technology (GT), North Carolina A&T University (NCAT), University of California Los Angeles (UCLA), and University of California Riverside (UCR) (Fig. 1).

We conducted our analysis using a modified version of the ***sock*** program [3]. TCP data transfers were generated. The transfer duration was varied by modifying the file size. Each node, at a specified time, assumed the role of client and server by executing several instances of ***sock.*** The ***sock*** program was modified to collect desired information of a particular flow: IP addresses, port numbers, block size, and arrival time. Once the scenario was completed, the data was copied to a central location for later analysis.

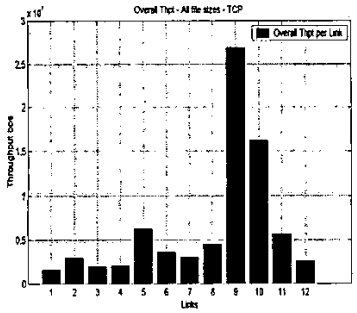

Fig. 2. Average throughput of each path.

Table 1. Link descriptions.

Link #	Link Description
Link 1	UCR->NCAT
Link 2	GT->NCAT
Link 3	UCLA->NCAT
Link 4	UCR->GT
Link 5	NCAT->GT
Link 6	UCLA->GT
Link 7	NCAT->UCR
Link 8	GT->UCR
Link 9	UCLA->UCR
Link 10	UCR->UCLA
Link 11	GT->UCLA
Link 12	NCAT->UCLA

C. *Results*

Fig. 2 gives the average throughput over the experiment duration seen by TCP flows over each path. Link 9 appears to have the greatest average throughout, while Link 1 is the slowest link. This study reaffirms what many before it have concluded; the Internet suffers heavily from load imbalances and even link asymmetry along the same path. This is a result of link capacity imbalance as well as variable demands placed on different links at various times.

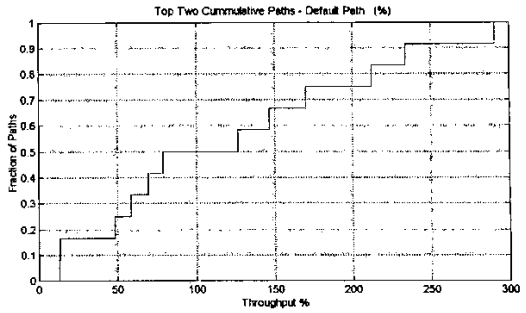

Fig. 3. Top two cumulative paths - default path (%).

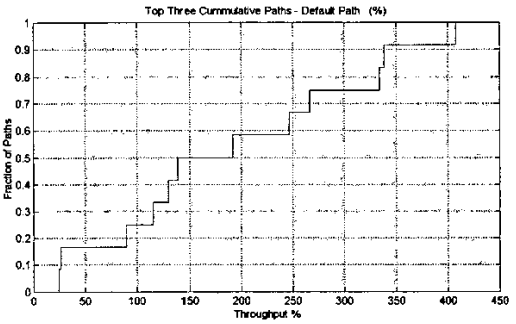

Fig.4. Top three cumulative paths - default path (%).

D. *Alternate Paths*

Until the work done in [1], most literature in the traffic measurement community stopped at that conclusion drawn in *Section II. C.* However, in [1], Savage et al. discussed how, if underutilized links were compounded to create alternate paths to a destination, a significant portion of the flows would have an alternate path with a higher capacity than the default path.

In [17], the authors used this idea of rerouting over alternate paths in their Detour project. They describe an architecture and provide promising results that were obtained using a testbed in a lab with traffic generators.

The authors of [18] and [19] went even further in that they implemented traffic rerouting in actual WAN testbeds. In the most recent work [19], the authors' results show that 70% of the rerouted flows on the WAN testbed experienced a 30% - 120% increase in throughput over the average throughput of the default path. These studies show firsthand the benefit of using traffic rerouting on the Internet.

E. *Aggregate Throughput over Alternate Paths*

Based on the results from *Section II. C.* (average throughput of each link), using the existing testbed with four

nodes and a total of 12 paths (every node is a source and destination node), we synthetically construct alternate paths for each source node to each destination node. We now synthetically stripe the data along the best paths. This could be a combination of the default and alternate paths or several alternate paths, depending on the path quality. The results here are a best case, and assume the data is intelligently divided at the sender to make the reordering for the application layer trivial. Fig. 3 is a CDF of the throughput of the best *two* paths combined, minus the throughput of the default path, for each source to each destination. Similarly, Fig. 4 is a CDF of the throughput of the best *three* paths combined, minus the throughput of the default path, for each source to each destination. We observe that each node would significantly benefit from striping. When striping over two paths (Fig. 3), we see an increase over the average throughput of the default path ranging from 10% to 280%. Accordingly, we see an increase ranging from 25% to 400% when synthetically striping over three paths (Fig. 4). Though this synthetic construction of alternate paths was done offline, it shows firsthand the extent of underutilized links (over time), and motivates the real-time use of ALST.

III. GENERALIZED APPLICATION LAYER OVERLAY

A. Background

An overlay architecture is an architecture where the current infrastructure remains in place and a virtual infrastructure or network is run atop it. Overlay networks are often the approach of choice, given that they can provide instant "results" and can span multiple autonomous systems without agreement or cooperation of the ISPs, thus avoiding their logistical conundrum.

Overlay architectures have been used in various instances; ranging from mobile networks [7, 8], virtual private networks [9], computer virus enabling, peer-to-peer file sharing networks [10], and probably the most popular, end system multicast [4, 5].

B. GALO Logical Architecture

Thus far, we have given an overview of QoS and QoS-based routing. We have also presented an empirical study of Internet traffic dynamics. The throughput measured in this empirical study of the WAN links is extracted and used as motivation for additional work. We now propose an overlay architecture that will provide the backbone for an immediately deployable QoS-based routing scheme. In accordance with past overlay approaches [4, 7, 8, 9, 10], this scheme requires no modification to intermediate nodes (routers and switches) and requires only a software upgrade to collaborating end nodes (workstations). The software upgrade is minimal in that a user can install the program without root privileges or any kernel modifications. The Generalized Application Layer Overlay (GALO) architecture is designed in a modular fashion and has three primary components: Distributed Client Engine (DCE), QoS Routing Engine (QRE), and the Forwarding Engine (FE). The FE is optional depending on the extension modules. We plan to use the data gathered by GALO to perform Application Layer Striping which requires the FE functionality.

- ***DCE:*** The DCE is a process that resides on each collaborating node. The primary purpose of the DCE is to transmit path quality updates back to the QRE. The DCE continuously calculates path quality depending on the specified metric. In our case, throughput is the only metric considered. Depending on the desired level of path quality accuracy, the DCE varies the rate at which it transmits control packets to the QRE.

- ***FE:*** The FE resides collocated with the DCE on each collaborating node. The FE is invoked only at transit nodes. Its primary responsibility is to act as the switching engine for passing traffic. Once the appropriate signaling[2] is received from the QRE, the FE dynamically modifies its engine to allow incoming traffic to be received and switched along the outgoing path. The FE's strength lies in its modular design. The current specification requires that the actual switching take place at the application layer. This approach has an obvious inefficiency in that the traffic must traverse the protocol stack to the application layer to be switched, and then traverse the stack again to continue to its destination. As such, this approach initially appears inefficient. Though the success of this technique is a factor of 1) hardware capacity; 2) processing load of the machine; 3) number of hops for a rerouted flow; 4) as well as the inefficiencies of traversing the entire protocol stack twice; we show (in Section V) that this delay will be negligible and a significant amount of the striped flows greatly benefit from this technique. Also, this approach is desired because it is directly in line with our goals of no modification to the intermediate infrastructure and minimum modification to the end nodes. Another possible approach is to use Source Routing between nodes. This would require that each collaborating node support Source Routing and would be more efficient, as the traffic would only have to go to the network layer for routing decisions, before being sent back out to its next hop. This approach would prove more efficient, but would require more invasive software modification (recompiling the kernel).

- ***QRE:*** The QRE is centralized in our design and is considered the brains of the architecture. It is the controlling unit for each external process and is responsible for every flow that traverses the network within this domain. The primary responsibility of the QRE is to maintain a current and accurate picture of the path quality between the collaborating

[2] The Application Layer Communication Protocol (ALCP) is the corresponding signaling/communications protocol that was designed to support this architecture. Due to space limitations, the description of the ALCP is not presented.

nodes. This is achieved by collecting and analyzing the UPDATE messages sent from each QRE and maintaining a table that contains accessibility information to each node. A modified version of Dijkstra's algorithm is used as the table update algorithm. Once the path quality of each link is known, the QRE has the capability to generate control messages and signal to the appropriate nodes specific information, depending on the application running atop GALO.

Available bandwidth estimating techniques can be classified into two categories: passive measurement [11, 12] and active probing [13, 14, 15]. Passive measurement tools use the trace history of existing data transfers. While potentially very efficient and accurate, their scope is limited to network paths that have recently used passive probing and are best initially deployed in an environment where end nodes communicate regularly, allowing passive non-invasive sampling. Therefore, in accordance with our primary design goal of an immediately deployable solution, GALO uses the passive measurement paradigm across a network whose end nodes communicate regularly. One such environment is that of a geographically diverse corporate extranet with mail servers that have recurrent communication. In our model, to allow bandwidth measurements, we emulate the aforementioned environment and create our own data to passively sample.

To obtain the path quality, artificial traffic is generated on each link using a modified version of the sock program [3]. Each node (DCE) has specific ports dedicated to capturing traffic to generate the path quality measurement, namely, in our case the current throughput. In addition to measurement sockets, several control stream sockets are reserved for messages between the DCE and QRE.

IV. APPLICATION LAYER STRIPING

A. Background

Striping is the concept used to combine multiple resources transparently to obtain a higher level of performance [20]. Two primary areas where striping has proven beneficial are network systems [20] and in data archival and retrieval [21].

Striping in network systems at multiple layers has been deployed in the IP, as well as in the ATM worlds. In [22], the authors discuss, among other things, the benefit of ATM cell striping across several SONET frames. In RFC 2950 [23], the Stream Control Transmission Protocol (SCTP) is presented and can provide striping across one or more interfaces. References [24] and [10] present several Application Layer approaches to stripe data (along the same path) to improve throughput and to attain maximum utilization of network resources. Striping has also been used to improve throughput in wireless networks when data is striped over multiple wireless interfaces [26], as well as in satellite systems to alleviate the TCP window size problem in links with a long delay [25].

Disk striping is a concept used to spread data across multiple disks. This concept is not new, but has been used for many years. Disk striping is available in two forms: single-user and multi-user. Single-user striping is of interest to us and will be discussed further. Single-user striping allows data broken into specific units (labeled as the stripe width) to be written in parallel to multiple disk drives [21, 27]. Thus, the average transfer rate, theoretically, can be improved proportional to the number of drives. This is useful when transfer time (not head movement) is the bottleneck.

The striping concept addresses a general question: Can multiple resources be used in parallel to increase a specific performance metric? In the single-user disk striping scenario, the resources are disk drives, and the parameter of interest is transfer time. In this paper, we further extend this notion to the Internet. In such an extension, the resources are specific paths to a single destination and the metric of interest is transfer time or overall (cumulative) throughput. The stripe width could be chosen at the application layer and the data could similarly be striped across different paths to increase the cumulative throughput.

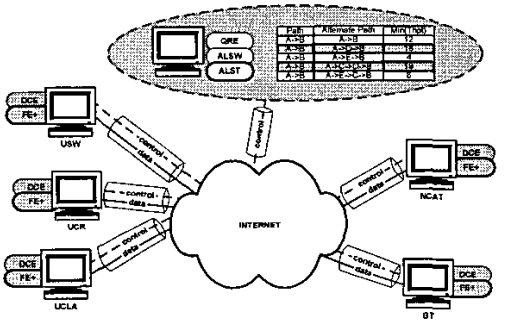

Fig. 5. GALO Architecture extended to support Application Layer Striping.

B. Overview of Application Layer Striping

Application Layer Striping (ALST) is a concept introduced to improve the traditional, best-effort service provided by the Internet. Several techniques for striping have been proposed [10, 20, 22-26], however none consider striping in the context of QoS-based routing. Further, most only consider streaming data over the same logical path. ALST alone is not considered QoS-based routing; but, when one considers the QoS of the links across which the data will be potentially striped, ALST falls into the category of QoS-based routing. The basic concept behind ALST is to use multiple paths in parallel along the Internet to carry portions of data destined for a certain node. This is achieved by constructing non-shortest path routes through intermediate nodes by appending different link combinations from the source node to end node. Fig. 6 shows an example of a flow being striped over two higher bandwidth alternate paths.

ALST uses an overlay approach where no modification of intermediate nodes is necessary. We propose an extension to GALO, the Application Layer Striping Module. The more end nodes, contained in GALO, and the more geographically diverse they are, the greater the probability of alternate paths on which to stripe data. The accessibility table generated in the QRE is also passed as input to the ALST module where, based on the specified metrics (we only consider throughput), alternate paths are created by appending multiple links. Once the quality of the alternate paths is known and the number of paths to stripe has been decided (because of fairness we currently consider a maximum of three), the process begins. The PROVISION message is signaled to the corresponding FE within the path nodes to provision the forwarding engine in each node. Also, the source node and the destination node are signaled to specify where to send the traffic and from where to expect the traffic, respectively. Additionally, the source is given instructions on how to split the data before sending, and the receiver is given instructions on how to reassemble the data. The data is sent shortly thereafter. Alternatively, we can use an intelligent transport mechanism (i.e., pTCP [28]).

C. *Application Layer Striping Software Architecture*

The input to the ALST module is the path quality table that is populated by the QRE. The ALST module monitors the shared memory segment containing the path quality table. The module then queries the path quality table and uses the data to generate the optimal alternative paths (a maximum of three), and thus makes the appropriate decision to stripe a flow. The optimal path algorithm is a variation of the popular Dijkstra's algorithm. The decision is passed back to the QRE, and the QRE messages to the respective DCE to split the data accordingly and to stripe the next flow.

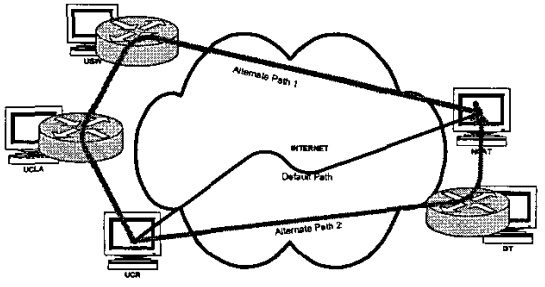

Fig. 6. Example of a flow rerouted and striped using Application Layer Striping.

D. *Application Layer Striping Logical Architecture*

To accomplish ALST, we propose an extension to our proposed Generalized Application Layer Overlay, the Application Layer Striping module (Fig. 5). For ALST to be feasible, we use GALO, which assumes that an abundance of trusted, cooperative end nodes (i.e., machines onsite at different locations of a major corporation) are in place. The more end nodes and the more geographically diverse they are, the more paths to choose from, which increases the likelihood of a more desired path. Once the grid of supporting nodes is constructed, path discovery and traffic sampling begins. Periodic updates are broadcast to the QRE using the Application Layer Communication Protocol (ALCP), where it constructs and maintains an accessibility table. This accessibility table is input to the ALST module where, based on specified metrics (we only consider throughput), alternate paths are created by appending multiple links. Once the path tree is constructed, an optimal path other than the default shortest path can be chosen if it meets the desired improvement in path quality. Once a better path has been defined, the PROVISION message is signaled to the FE on the corresponding path nodes to provision the forwarding engine in each node. These intermediate forwarding nodes act as switches throughout the duration of the rerouted flow (Fig. 6). Also, the source node and the destination node are signaled to specify where to send the traffic and where to expect the traffic, respectively. The data is sent shortly thereafter. The data maintains the path until a threshold has passed, a shorter more direct path with acceptable path quality is found, or another path (possibly more hops) is located with a more desirable path quality.

V. PERFORMANCE ANALYSIS

A. *Experimental Setup*

We expanded the initial network discussed in Section II to have a total of five end nodes[3] and an additional node used to house the QRE. All workstations are shared, general purpose machines. Four of the five end nodes are running Linux, both the fifth end node and the node that houses the QRE have Solaris as their operating systems. The default file size of each flow is 100KB. If data is rerouted over two paths, the flow is statically split in half (half of the default flow 100KB – 50KB per path). Likewise, if the flow is routed over three paths, each flow is approximately 33KB. To perform the experiments in an efficient manner, we designate two separate blocks.

The sampling block is the period of time that a probe flow[4] is sent from each source to each destination. Thus, creating traffic to sample, which allows the generation of link throughput data. Within the sampling block, probe flows (a total of 20 – one from each source to each destination) are generated and monitored. The probe flows run in series as to avoid friendly traffic interference. The second period is the reroute block. During this period, no probe flows are transmitted, only a rerouted flow. The QRE keeps a running average of each default path's throughput, and synthetically creates alternate paths. It then chooses the paths with the

[3] The fifth node is located at the University of Washington, Seattle.

[4] Again, the probe flows are only necessary for our experiments. Normally, this architecture would be deployed in an environment where passive sampling would be possible.

highest cumulative potential increase as the primary candidates for striping. During the next reroute cycle, the chosen candidate is rerouted over the alternate path. The alternate paths have been limited to a maximum of two hops. Fig. 6 is an illustration of rerouted flow having two hops.

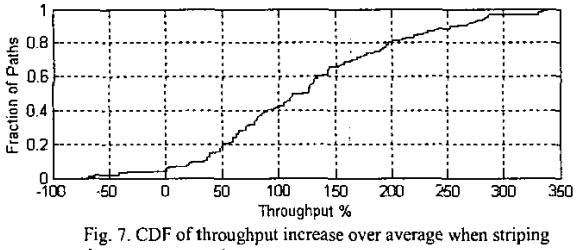

Fig. 7. CDF of throughput increase over average when striping data over two paths.

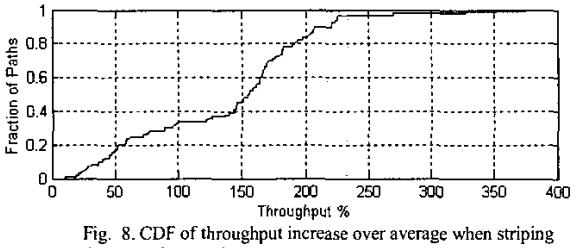

Fig. 8. CDF of throughput increase over average when striping data over three paths.

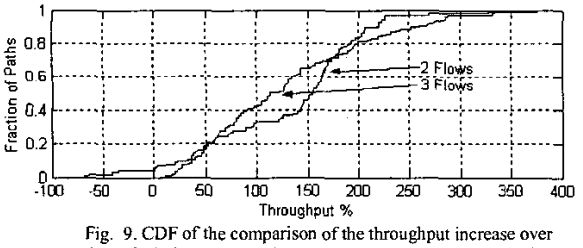

Fig. 9. CDF of the comparison of the throughput increase over the default throughput, when striping over two and three paths.

In order to efficiently test this architecture, we repeated the periods throughout the length of the experiments. Thus, the entire duration of the experiments consisted of the repetition of a sample block followed by a reroute block. The default time for both blocks was 130 seconds. Accordingly, every 260 seconds, a new flow was rerouted and the average throughput of the default links was updated. As a result of limited machine access, the code was executed in user space. Over the duration of the experiments approximately 100 striped flows were generated for each scenario.

B. Results

The results for ALST are shown in Figs. 7 - 9. Again, this is a best case scenario that assumes that the data is initially segmented at the sender using an intelligent striping protocol so that reassembly at the receiver is trivial. Additionally, these results do not take into account that the flow striped along the slower path will degrade the overall throughput because the aggregate throughput shown in the graphs is only valid when both flows are being received simultaneously. It is believed that this minimal delay will not destroy the integrity of the technique and that the flows, overall, will still benefit significantly from striping.

B.1 Two Striped Paths

Fig. 7 shows the aggregate throughput of a 100KB flow being striped over the top two logical paths compared to the average throughput seen by the default path. We observe that over 90% of the flows benefit from this technique extending up to approximately 340%.

B.2 Three Striped Paths

Fig. 8 illustrates the aggregate throughput of a 100KB flow being striped over the top three logical paths compared to the average throughput seen by the default path. We observe that all of the flows benefit from this technique with several flows exceeding 350%.

B.3 Two Striped vs. Three Striped Paths

Fig. 9 compares the aggregate throughput increase over the default path of the flows striped over two and three paths. Surprisingly, both streams perform about the same with the striping over three paths performing marginally better. This behavior is a result of the overhead encountered with increasing the number of application layer switched paths. Additionally, the flows over the three paths were only 33KB opposed to 50KB, with the smaller file size having a less likely chance of taking advantage of TCP's maximum window size. Thus, it is believed that the data rate is still proportional to the number of paths over which the data is striped. However, we find that the scheme is also significantly affected by overhead encountered along the application layer switched path, as well as the size of the striped flow, thus, virtually offsetting the deficiencies and gains.

VI. CONCLUSION & FUTURE WORK

In this paper, we proposed a deployable approach to improving QoS, by using the GALO overlay architecture along with the ALST extension in order to immediately, in a peer-to-peer approach, improve QoS for traffic flows. Considering fairness, we limited the number of striped paths to a maximum of three. We show that by using the striping technique, the flows achieve a maximum throughput increase over that of the average throughput of the single default path, with a maximum increase exceeding 350%.

REFERENCES

[1] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson, "The End-to-End effects of Internet path selection," in Proceedings of the ACM SIGCOMM, Oct. 1999, vol. 29, pp. 289-299.

[2] V. Paxson, "End-to-End Internet packet dynamics", ACM SIGCOMM '97, September 1997.

[3] W.R. Stevens, TCP/IP Illustrated Volume 1: the protocols. Boston, MA: Addison Wesley, 1994.

[4] H. Erikson. "MBONE: The Multicast Backbone". Communication of the ACM, pages 54–60, August 1994.

[5] Macedonia, M. R., Brutzman, D. P., "MBone provides audio and video across the Internet," IEEE Computer, Vol.27 #4, April 1994, pp. 30-36.

[6] How to Debug an MBone Session. http://www.informatik.unimannheim.de/informatik/pi4/projects/Cost264/HowToDebugMBone.html. April 2002.

[7] M. Stemm, "Vertical Handoffs in Wireless Overlay Networks," Master's thesis, UC Berkeley, May 1996.

[8] R. Katz and E. Brewer. "Wireless Overlay Networks and Adaptive Applications." In the Proceedings of MobiCom 1996.

[9] VPN Overlay Networks: An Answer To Network-Based IP VPNs? http://networkmagazine.com. February 2002.

[10] Stephanos and Androutsellis-Theotokis. White Paper: A Survey of Peer-to-Peer File Sharing Technologies. ELTRUN, Athens University of Economics and Business, Greece.

[12] M. Stemm, S. Seshan, and Randy H. Katz. "A network measurement architecture for adaptive applications." In the Proceedings of IEEE Infocom 2000, Monterey, CA, March 2000.

[13] B. Mah. pchar: A tool for measuring internet path characteristics, 2001. http://www.employees.org/ bmah/Software/pchar/.

[14] V. Jacobson. pathchar - a tool to infer characteristics of internet paths, 1997. presented as April 97 MSRI talk.

[15] N. Hu and P. Steenkiste. Estimating Available Bandwidth Using Packet Pair Probing. September 9, 2002 CMU-CS-02-166. Technical Report.

[16] A. Ghosh, M. Fry, and J. Crowcroft. "An Architecture for Application Layer Routing." In the Proceedings of IWAN 2000

[17] S. Savage, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vahdat, G. Voelker, and J. Zahorjan, "Detour: a Case for Informed Internet Routing and Transport," IEEE Micro, pp. 50-59, v 19, no 1, January 1999.

[18] D. Andersen, H. Balakrishnan, M. Kaashoek, R. Morris, "Resilient Overlay Networks," Proc. 18th ACM SOSP, Banff, Canada, October 2001.

[19] R. Beyah, R. Sivakumar, and J. Copeland. "Application Layer Switching: A Deployable Technique for Providing Quality of Service." To appear in the proceedings of Globecom 2003, December 2003.

[20] C. Brendon, S. Traw, and J. Smith, "Striping within the network subsystem," IEEE Network, vol. 9, no. 4, pp. 2232, July 1995.

[21] Disk Striping.http://www.webopedia.com/TERM/D/disk_striping.html. April 2002.

[22] J. Duncanson, "Inverse multiplexing," IEEE Communications Magazine, vol. 32, no. 4, pp. 3441, Apr. 1994.

[23] R. Stewart et al, "Stream control transmission protocol," IETF RFC 2960, Oct. 2000.

[24] H. Sivakumar, S. Bailey, and R. Grossman, "PSockets: The case for application level network striping for data intensive applications using high speed wide area networks," in Proceedings of IEEE Supercomputing (SC), Dallas, TX USA, Nov. 2000, pps. 240-245.

[25] M. Allman, H. Kruse, and S. Ostermann, "An application-level solution to TCP's satellite inefficiencies," in Proceedings of Workshop on Satellite-Based Information Services (WOSBIS), Rye, NY USA, Nov. 1996.

[26] H.-Y. Hsieh and R. Sivakumar, "A transport layer approach for achieving aggregate bandwidths on multi-homed mobile hosts." ACM MOBICOM, Atlanta, GA, September 2002.

[27] Disk Striping as a Performance Enhancement Tool for Multi-User (UNIX) Systems. http://www.1776soft.com/striping.htm. February 2002.

[28] H.-Y. Hsieh and R. Sivakumar, "pTCP: An end-to-end transport layer protocol for striped connections." In the Proceedings of ICNP 2002.