Improving VPN Performance over Multiple Access Links

Jack Brassil, Rick McGeer, Raj Rajagopalan, HP Laboratories, *Andy Bavier,* Princeton University
Larry Roberts, Anagran Inc., *Brian Mark,* George Mason University, *Stephen Schwab,* Sparta Inc.

***Abstract*— To improve the performance of VPN connections we investigate how the bandwidth of multiple access links can be aggregated with inverse multiplexing to create a single, higher capacity logical communication link. But achieving the maximum possible aggregated TCP throughput becomes extremely challenging if the underlying links either use different technologies (e.g., DSL, cable modem) or suffer different or time-varying communication characteristics (e.g., available bandwidth, packet loss rate).**

To maximize VPN throughput we have constructed a system that combines two distinct innovations. First, we continuously measure the communication characteristics of the underlying component links in our aggregate and dynamically assign packets to each link in proportion to its available capacity. Second, we modify TCP congestion control across the inverse-multiplexed access hop to avoid rate decreases normally initiated by the delayed acknowledgments often triggered when using legacy TCP on multiple heterogeneous paths. We describe the system's implementation, the test environment we built on *Emulab*, and show that when access links form the communication bottleneck in the end-to-end connection we can significantly increase VPN performance over conventional approaches. [1]

I. Introduction

Many Small Office/Home Office (SOHO) workers are required to use enterprise-wide Virtual Private Network (VPN) software to access enterprise networks securely. A common complaint heard from VPN users is that remote access performance is often poor. More precisely, users complain about slow connection establishment, relatively poor throughput when communicating to enterprise systems (relative to access from within the intranet), and reduced internet access throughput via the VPN (relative to direct internet access).

One of the many reasons VPN-based systems suffer low throughput performance is that access links continue to remain a bandwidth bottleneck in the end-to-end connection. VPN tunneling often restricts packets to follow a single path, even in cases where multiple access links are available. Yet with falling costs the prevalence of multiple internet access links at a single site (or residence) is growing; in some cases to maximize security and minimize enterprise liability users are required to maintain separate links; one for enterprise access and one for internet access or personal use. In many such cases one link remains idle even while the second link serves as the choke point for the active VPN connection. But for those cases where additional links may be used, we have developed new techniques to permit users of existing, unmodified VPN software to use multiple links to improve their performance.

Our system uses inverse multiplexing at the IP layer to aggregate access links. Inverse multiplexing is a conventional technique for aggregating the bandwidth of multiple communication links, and it can be performed at any of several protocol layers (e.g., physical, link, network). In conventional settings, the underlying links used have been similar or identical (e.g., bonding of multiple T1s). But the increasing prevalence and lowered cost of diverse access network technologies (e.g., DSL, cable, fiber) compel us to investigate the performance of aggregating the bandwidth of links with dramatically different communication characteristics to boost VPN performance.

Towards this goal in an earlier paper we introduced *NATALIE* [1], a **N**etwork-**a**ware **tra**ffic equa**lize**r. NATALIE combines an arbitrary set of network interfaces and schedules IP packet transmissions over those interfaces in a weighted round-robin (WRR) fashion. To maximize throughput NATALIE measures the communication characteristics of the underlying links and dynamically adjusts its scheduler to assign packets to each link in proportion to its available capacity.

VPN software can typically be configured to use either TCP or UDP. To address a common and challenging VPN performance problem we focus on the case of a single TCP flow, and how that flow behaves when

[1]This work was supported in part by DARPA Contract N66001-05-9-8904.

split over multiple, heterogeneous links. Considerable previous research [2], [3], [4] has shown that inverse multiplexing a single TCP connection over heterogeneous links to obtain aggregated bandwidth is difficult; one can naively attempt to combine the bandwidths of 2 different links and realize even less bandwidth than the bandwidth of the slower of the two links [5]. Splitting TCP segments belonging to a flow can and does result in out-of-order packet delivery that can significantly decrease TCP throughput; sufficiently misordered packets trigger timeouts and unnecessary retransmission requests by the TCP receiver. In our earlier paper we have showed that using NATALIE's WRR algorithm can substantially increase the diversity of the underlying links where gains can be achieved.

Nonetheless, even with adaptive WRR scheduling if we allow the component links to have sufficiently different characteristics (e.g, propagation delay), legacy TCP itself becomes the performance barrier. The TCP sender congestion window limits how fast a sender can inject packets into the network, and consequently limits throughput. The window size is adjusted in response to measured network conditions. When an out-of-order packet arrives, the TCP receiver generates a duplicate *Ack* with a sequence number that has been acknowledged previously. If the number of duplicate acks reaches a fixed threshold (e.g., 3 packets), a sender using *Fast Retransmit* infers that a packet was lost and retransmits it. Therefore, transmitting packets from the same TCP connection over paths with different delays misleads the sender into shrinking the congestion window unnecessarily, reducing throughput dramatically. TCP Selective Acknowledgment (*SACK*) allows more detailed feedback of which out-of-order packets are received and hence reduces unnecessary retransmissions. However, *SACK* does not address the problem caused by loss rate variation across different links.

Hence, as a result of the design of conventional TCP congestion control mechanisms, VPN traffic is often unable to take advantage of bandwidth otherwise made accessible by multi-path networking [17]. In this paper we present the design of a system that combines both inverse multiplexing and a new TCP congestion control mechanism – *TCP-Explicit Rate* or *TCP-ER* – that is better suited for operation over multiple paths. We focus on the challenging problem of splitting a single TCP connection across multiple, heterogeneous links – a common VPN configuration. We note that several other research groups have shown that modifications of TCP can help maintain performance in multipath environments [24], and some of these approaches are likely to be useful in the VPN setting we consider as well.

The remainder of this document is organized as follows. Section II presents the design and implementation of the traffic equalizer, the modified TCP congestion control in TCP-ER, and the testbed used to develop our software and test its performance. The next section describes the empirical system performance when the characteristics of the underlying communication links are known. Section IV discusses observations based on our performance results and identifies approaches to measuring the parameters of links in our aggregate. Section V examines related work, and the final section summarizes our work.

II. Background

A. NATALIE

NATALIE is based on a modified version of a Linux-based traffic control kernel module called Traffic Equalizer (TEQL) included in modern Linux distributions. TEQL assigns each incoming packet to one of *N* physical network interfaces using a deterministic round-robin discipline. With NATALIE, the fraction of packets assigned to the each link (i.e., weight) can be assigned based on known parameters such as the underlying link bandwidths. NATALIE permits dynamic, real-time adjustment of link weights on the installed module by user-level programs. The module reads weight parameters from the /sys/module/NATALIE/parameter/weights directory, resets a packet counter for each link, and starts distributing packets according to the new weights. In this way, a user program, usually with root privileges, can dynamically adjust weights as network conditions change. Through this dynamic weighting NATALIE maintains high throughput as the controlling application measures the available bandwidth on each link, and adjust weights in proportion to bandwidth adjustments.

B. Testbed

Emulab [8] was chosen to test the performance of a TCP connection over multiple links with NATALIE. Emulab is a publicly available time- and space- shared network emulator, where arbitrary network topologies can be constructed, and link loss, latency, bandwidth can be easily user-defined and changed.

Figure 1 depicts our test topology where two end systems are directly connected with multiple links. Link emulation is performed by a node running *dummynet* that is transparent to the end systems. Each end system

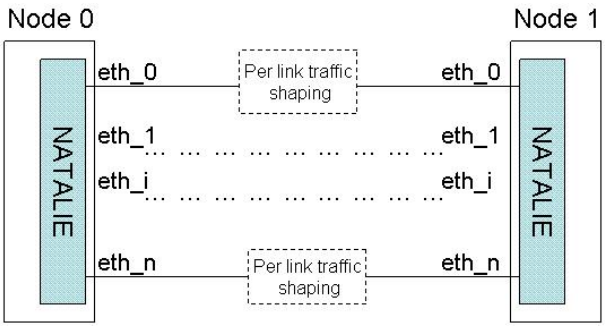

Fig. 1. A simple Emulab test topology.

is an Intel 64-bit Xeon 3.0 Ghz machine running the Redhat Linux Fedora Core 4 (2.6.11 kernel) operating system. These machines provide sufficient computing and I/O performance to ensure that our experiments are unaffected by background traffic generators and network measurement tools required for our experiments. We set the tcp_no_metrics_save option to ensure memoryless TCP operation from experiment to experiment. Though we are focused on access link settings where edge propagation delays are typically small, even small delays must be considered for performance optimization. We anticipate considerably higher transmission speeds in future access networks, and our aggregate transmission speed in our experiments is up to 300 Mbs. Hence even access networks with propagation delays as low as 5 ms. require us to increase TCP memory size to ensure that the sending window size was not limited by the system bandwidth-delay product. In each of our experiments we set the following capacities at both sender and receiver:

$$net.core.rmem\_default = 111616$$
$$net.core.rmem\_max = 1310710$$
$$net.core.wmem\_default = 111616$$
$$net.core.wmem\_max = 1310710$$
$$net.ipv4.tcp\_wmem = 4096\ 16384\ 1310720$$
$$net.ipv4.tcp\_rmem = 4096\ 87380\ 1747600$$

While setting per connection maximum window sizes to approximately 1 MB does not guarantee that our system will not be window bound, it does not appear to have been a constraint in any of the results presented here.

C. TCP-ER: Explicit Rate-Aware TCP

TCP-ER is a modified TCP stack that differs in two significant ways from a conventional TCP stack. First, the system implements an end-to-end QoS signalling protocol to communicate rate information between end systems (e.g., source and receiver) and network routers (if present). Second, a modified TCP congestion control algorithm allows a sender to exploit knowledge of internal network behavior (as communicated via the signalling protocol) to send at an instantaneous rate that can be supported by the network.

TCP-ER was originally developed to operate in conjunction with explicit rate capabilities of the Anagran FR-1000 flow router [6]. In addition to operating as a conventional IP best-effort router, the FR-1000 provides a number of special services to enhance the quality of service offered to clients equipped with TCP-ER protocol stacks. The router natively identifies, classifies and routes flows. It also supports a number of advanced traffic management capabilities, including the ability to offer transport types other than traditional best-effort traffic, including *available rate* and *maximum rate* service.

Anagran routers provide *explicit rate* feedback to clients. That is, a router continuously informs clients of the instantaneous bandwidth available for them to use on each active connection. A client equipped with TCP-ER is capable of exploiting the availability of an end-to-end rate information sent through a network of flow routers. In contrast, a legacy TCP client dynamically uses TCP state information to infer the bottleneck capacity of an end-to-end path, and adjusts its transmission rates to that available bandwidth.

TCP-ER uses an in-band signaling protocol known as TIA-1039 to explicitly establish the end-to-end QoS parameters required for each new flow. This protocol has been standardized by the TIA Satellite QoS group [7] as a preferred means to convey QoS parameters in an IP setting. Figure 2 depicts the structure of the header used in IPv6 to communicate rate information between end systems and routers.

We have developed a version of TCP-ER for X86-based PCs running the Fedora Core 4 operating system. TCP-ER congestion control supports a “fast-start” option invoked by a Linux ioctl() system call. When “fast-start” is invoked for a connection the TCP slow start algorithm is disabled, and the sender window size is instead determined by the rate specified by the received explicit rate parameters. This permits, for example, the sender’s transmission rate to immediately jump to the bottleneck bandwidth speed in a network in the absence of cross-traffic. Hence, very common short duration flows realize far better TCP performance than would be the case with a conventional system implementing

slow-start and congestion avoidance phases during rate increases. TCP-ER also exhibits excellent performance relative to legacy TCP in the presence of network impairments.

In networks where packet losses are frequent (e.g., wireless LANs) the sender receiving rate feedback can distinguish between loss due to congestion and loss due to link impairments and consequently not reduce rates in response to any loss event. Further, if rates are reduced by congestion then the slow-start phase of the rate recovery period can again be avoided. Avoiding slow start and congestion avoidance phases in the presence of high propagation delay (e.g., satcom) is particularly valuable, as it enables the sender to avoid idling during the long round-trip waits for transmissions to be acknowledged.

As with other conventional TCP stacks, the TCP-ER stack uses selective acknowledgment and retransmission (SACK) for error recovery on QoS-enabled connections. In the absence of ER information from a receiver (or flow router) the sender's TCP-ER stack simply operates in a conventional TCP congestion control fashion (i.e., with slow start). In a end-to-end connection using TIA-1039 signalling, both sender and receiver are required to run TCP-ER; no flow routers are required in the path. A more common configuration, however, is for end systems to make use of nearby *proxies* to provide the TCP-ER functionality. In this way enhanced TCP service can be incrementally added to a large number of end systems (e.g., all systems on a subnet or LAN) using any operating system and/or TCP stack without the need for the end system operating systems to be modified. In this paper, all test results we report are taken from tests where either source and destination TCP-ER based systems are connected with no intervening flow routers, or source and destination are each connected to a local proxy, and both proxies are connected without intervening flow routers.

We next describe how we envision combining the multipath routing of NATALIE with TCP-ER in a typical access setting. Let's consider the case of a single provider for access links. A multihomed router (or comparable customer premise edge device) is connected to 2 or more access links. The links are terminated in a single device at the provider's point-of-presence. Hence, the implementation is similar to a common configuration of hardware bonding technology. Since our system is single hop, a sender has visibility to all traffic on the hop. Hence, the sender can initiate a flow at any rate up to the total capacity of the aggregate links. We note that many other configurations are possible. For example,

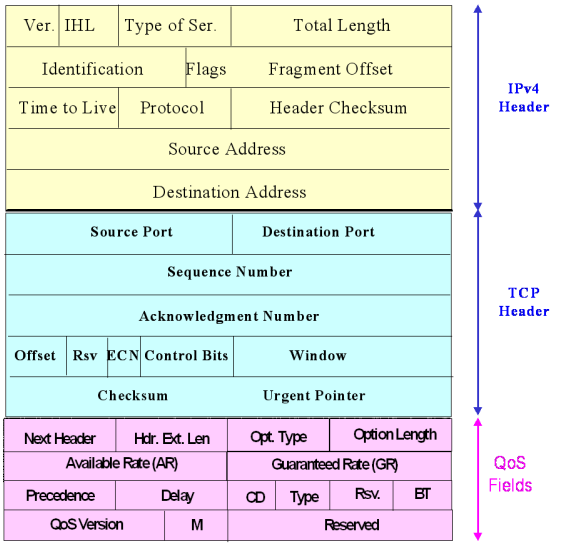

Fig. 2. IPv4 QoS signaling header extension.

in an asymmetric system where downstream bandwidth greatly exceeds upstream bandwidth, some applications (e.g., peer-to-peer file sharing) might desire to aggregate the upstream links but not aggregate the downstream. In such a configuration no provider equipment is required to terminate an aggregated connection. Indeed, all IP packets traversing different links are reassembled into a single TCP flow at the receiver in all cases. Hence, for the case where different access network providers are used, no provider equipment is required to support upstream aggregation.

III. Performance Evaluation

In this section we present throughput results for a single TCP connection using NATALIE to aggregate the bandwidth of three links on the topology of Figure 1. In particular we present a comparison of the performance of the system using *TCP-ER* versus *TCP-BIC*, the default TCP congestion control algorithm in our FC4 systems. In each experiment communication link parameters (i.e., link transmission speeds, propagation delay, packet loss) are fixed, and no background traffic is present. All throughput measurements we present were obtained using Iperf v2.02 [11] for memory-to-memory transfer durations of at least 20 seconds. We have studied the operation of our system with the widely used *OpenVPN* [14] open-source VPN solution for Linux systems. In these tests we have consistently found that OpenVPN's reduction in file transfer throughput is quite negligible. Hence we have chosen to exclude its use in the set of experiments reported here. We are keenly

aware, however, that poorly written or misconfigured VPN software, as well as overloaded VPN servers are the basis of a significant amount of the performance problems reported by users. We will leave the study of VPN software and operational systems for future research work.

In the remainder of this section we seek to answer questions such as the following: How much higher throughput performance can we expect using TCP-ER rather than legacy TCP in a multi-link access setting? How should link weights be set to maximize performance across links? How much more diversity in component link parameters and characteristics (e.g., transmission speed, packet loss) can be tolerated while still achieving aggregation gains in TCP-ER?

Let's begin with the simplest case, where each of the 3 component links have identical communication parameters. Let the link weights be $w_i,\ i = 1, 2, 3$, and the fraction of packets assigned to the i^{th} link be $\frac{w_i}{\sum_{i=1}^{3} w_i}$. Also, let $T(w_1, w_2, w_3)$ be the steady-state TCP throughput achieved by the system with weights w_1, w_2, w_3.

Table I shows the throughput performance of a legacy TCP connection split using NATALIE over 3 100 Mbs links, each with no packet loss and negligible propagation delay. We consider weights in the set $w_i \in \{2, 3, 4\}$. Note that if the 3-tuple of weights is $< 2, 3, 4 >$ the fraction of packets assigned to each of the links are $\frac{2}{9},\ \frac{1}{3},\ \frac{4}{9}$, but if the 3-tuple of weights is $< 2, 4, 4 >$ the fraction of packets assigned to each of the links are $\frac{1}{5},\ \frac{2}{5},\ \frac{2}{5}$. Each link is independently capable of achieving a throughput of approximately 94 Mbs, so we can estimate that the aggregate of three links would have a maximum possible throughput of approximately $3 * 94$ Mbs $= 282$ Mbs. As expected, this result is very nearly achieved by $T(2, 2, 2)$, where links are equally weighted.

Note how throughput performance is sensitive to link weights; for example, T(2,2,4) is about one-third of T(2,2,2), indicating that improperly weighting even a single link by a factor of 2 can considerably reduce aggregate throughput. As we will see this sensitivity only increases when links suffer impairments such as delay and loss.

Now let's consider an aggregate of three links with bandwidths of 100, 75 and 50 Mbs. Suppose we add a small unidirectional packet loss rate (0.005) and a small propagation delay (5 ms.) to each link; though these transmission rates are high for access links the loss and delay might roughly correspond to what would be found if access links were of similar technology (e.g., DSL) and offered by a single access provider. We will assume that the packet losses are independent from packet to packet and link to link, and take this to be the case throughout the remainder of our work.

w_1	w_2	w_3	Throughput (Mbs)
2	2	2	280
2	2	3	220
2	2	4	188
2	3	2	220
2	3	3	251
2	3	4	211
2	4	4	235
3	3	4	235
3	4	3	234
3	4	4	258

TABLE I

THROUGHPUT FOR 3 AGGREGATED 100 MBS LINKS AS LINK WEIGHTS CHANGE.

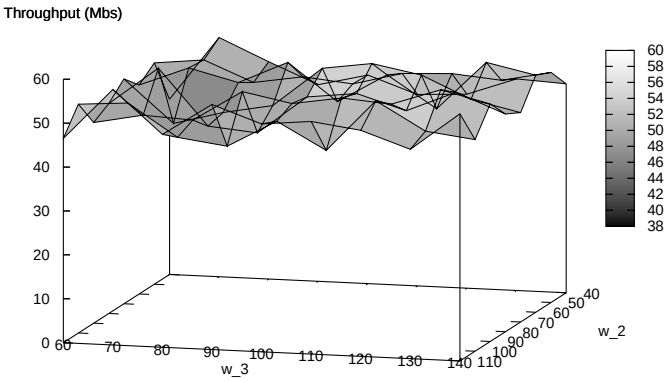

Fig. 3. The throughput of 3 aggregated links with speeds 50, 75, 100 Mbs using legacy TCP with fixed weight $w_1 = 50$ and variable weights w_2 and w_3.

Figs. 3 and 4 depicts the throughput performance of our aggregated links when using legacy TCP and TCP-ER with a fixed weight $w_1 = 50$ while varying weights w_2 and w_3. The first and most striking observation is how dramatically TCP-ER's aggressive congestion control permits considerably higher throughput than legacy TCP for nearly all values of weights. Since each link shares a common set of impairments, as expected we find the throughput under TCP-ER is maximized near values of $w_2 = 70,\ w_3 = 90$, corresponding roughly to the transmission link speeds. Approximately

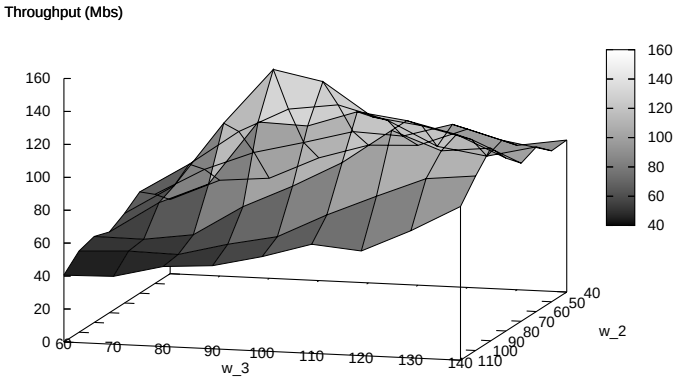

Fig. 4. The throughput of 3 aggregated links with speeds 50, 75, 100 Mbs using TCP-ER with fixed weight $w_1 = 50$ and variable weights w_2 and w_3.

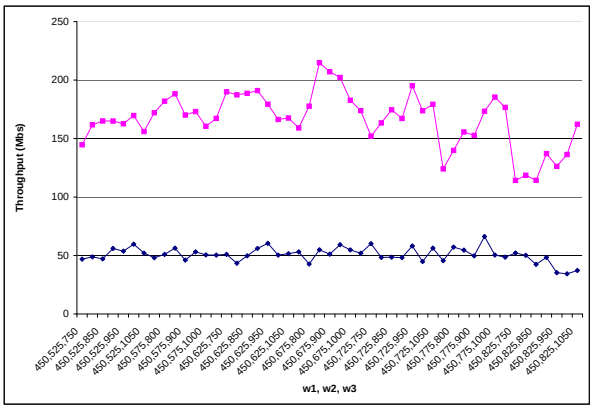

Fig. 5. The throughput of 3 aggregated links with speeds 50, 75, 100 Mbs using TCP-ER (upper) and TCP-BIC (lower) with varying weights. Each link has a unidirectional packet loss rate of .001 and propagation delay of 5 ms.

146 Mbs throughput is realized at these weights, despite the presence of a 0.1% round trip packet loss rate. The second notable observation is that TCP-ER throughput varies significantly as link weights change; indeed, in an extreme case throughput under TCP-ER can fall below throughput under legacy TCP.

Note that since the throughput surface for legacy TCP appears 'flatter' than for TCP-ER, one might be tempted to conclude that legacy TCP is less sensitive to varying weights that TCP-ER. This in fact does appear to be the case in most circumstances, and we speculate that is a consequence of the aggressiveness of the sender. But in the case of Fig. 3 the loss rate (0.1% round trip) is already high enough that TCP is able to achieve only a very limited degree of aggregated bandwidth, regardless of weights. Said another way, if we were to further increase the loss rate Fig. 4 would also appear to flatten. Of course, this flatter surface would rest at a much lower absolute throughput than shown here.

Fig. 5 presents a different view of a similar test case where we reduced the loss rate to 0.001. Here we focus a sliding window on throughput measured as we sweep all three link weights across a wide range of values. Again, we see the maximum throughput occurring near where the weights are in proportion to the link bandwidths (i.e., $< 450, 675, 900 >$). As we sweep across the link weights, we cross settings where the weights diverge considerably from these proportions, and these poorly selected weight settings correspond to the local minima on the chart. For example, the lowest throughput is achieved at $< 450, 825, 750 >$. But in ordinary circumstances we would not consider overweighting the 75 Mbs link while relatively underweighting the 100 Mbs link. In summary, for completeness Fig. 5 depicts a number of weightings we wouldn't normally permit, and this gives the appearance of sensitivity that we would not see in a well managed, operational system.

We next consider system throughput when aggregating heterogeneous links; that is, links with differing transmission speeds, loss rates, and propagation delays. The 3 links we studied are as follows: a 7 Mbs link with packet loss rate of .0050 and delay of 5 ms., a 6 Mbs link with packet loss rate of .0045 and delay of 8 ms., and a 5 Mbs link with packet loss rate of .0020 and delay of 7 ms. In this experiment the links speed and delays were sufficiently low that we maintained the TCP memory space at default system values without concern for being window-bound.

Given a mix of transmission speeds, delays and packet loss rates, it is often not immediately apparent which links offer the highest potential available bandwidth. When using either TCP or TCP-ER on each link *independently* the measured throughputs are approximately 6.7, 5.6, and 4.7 Mbs; their sum, 17.0, represents an upper bound on the aggregate throughput that could be achieved by aggregating all three links. Fig. 6 shows the resulting throughput for TCP-ER and legacy TCP. Again, TCP-ER outperforms at the expense of increased sensitivity to link weights.

IV. Observations

Proper selection of link weights is key to maximizing throughput over aggregated links. Further, we have

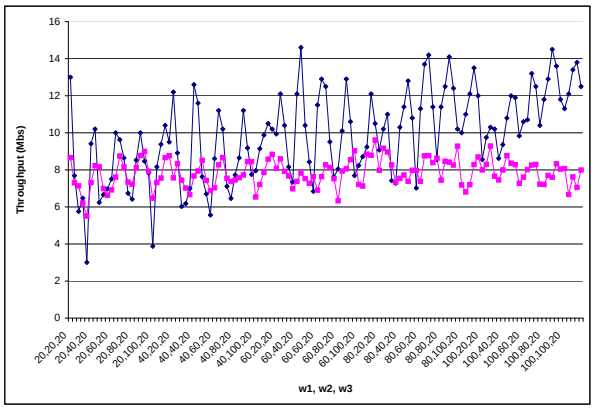

Fig. 6. The throughput of 3 aggregated heterogeneous links using TCP-ER (upper) and TCP-BIC (lower) with varying link weights.

shown that the sensitivity of throughput to this weighting can be higher in aggressive transports schemes such as TCP-ER relative to legacy TCP. It should be stated that, in general, we have not found this increased sensitivity to be a problem whatsoever in operating a number of system installations. Rather, the throughput increases made possible by TCP-ER are of considerable value. The appearance of high sensitivity in the experiments reported here is in part an experimental artifact; traffic variability that is normally found in actual operating environments due to background traffic, variable delays, etc, tend to result in throughputs that are closer to the average of those that can be achieved in the neighborhood of weight values near the actual setting.

In principle, identifying a preferred set of link weights is straightforward in static or quasi-static network environments; the available bandwidth of component links can be directly measured, and each link's weight can be set in proportion to its measured bandwidth. Measurements can be performed actively or passively. Active measurements typically rely on software tools to perform measurements, while passive measurements require observation of transport performance on each component link.

In dynamic settings, establishing time varying link weights can be quite challenging. This measurement challenge is well outside the scope of this paper. Nonetheless, if measurements are available, link weights can be changed at a rate that is limited primarily by the speed at which the file system can support a read/write operation.

An aggregation system using TCP-ER and its associated signalling protocol permits us one significant additional advantage that is not available in most environments. TCP-ER receives Explicit Rate notification from ER-capable routers in the path via the TIA-1039 signalling protocol. Hence, in a properly configured network topology, a sender can signal routers to obtain current available bandwidth on each of the component paths, and use this information to set link weights precisely. That is, having a router such as the Anagran FR-1000 in the upstream path of each link continuously conveying up-to-date Available Rate information for each path is far preferable to relying on end system based software available bandwidth measurement tools.

V. Related Work

Bandwidth aggregation has been attempted at every possible protocol layer. These approaches differ greatly in the system components that must be modified to achieve aggregation, the assumptions about the similarity of the underlying communication links, and whether the approach spans a single hop or an entire end-to-end connection. For example, at the link layer *bonding* has frequently relied on introducing hardware to combine multiple identical physical links. Such an approach has the advantage of not requiring changes to host protocols, while suffering from added component costs and establishing a relatively rigid configuration. One example of this approach is the hardware based *ethernet over copper* bonding product from Actelis Networks [13] that achieves symmetric access up to 70 Mbs over multiple DSL wire pairs using ITU Recommendation G.998 (G.Bond) [12].

A popular software-based technique used to logically combine serial data links by splitting, recombining and sequencing datagrams is *Multilink PPP* (MLPPP) as described in RFC 1990. Though not requiring new hardware, this approach is also intended to be used on multiple similar serial data links directly connecting two MLPPP-enabled routers or hosts.

In general, standards have not addressed the problem of aggregating dissimilar links, as we have studied in this paper. But the growing deployment of wireless networks – where in some cases even links relying on similar technology suffer dramatically different communication impairments – has stimulated investigations by a variety of researchers [16], [18].

Handling diverse links has focused attention on approaches to modifying TCP to support transmission over multiple channels [24], [21], [25]. These efforts attempt to address the inherent TCP assumption that packet misordering on a single path will be relatively infrequent, an assumption made invalid when using different links

for a single connection. The principle disadvantage of all such approaches is the requirement to modify the end system TCP stacks for all participating end systems, or – more commonly – to insert enhanced TCP proxies in the network path and route traffic from unmodified end systems through the proxies.

VI. Conclusion

As end systems have increased access to diverse and inexpensive access networks, we are faced with the challenge of using these communication links effectively in conjunction with existing VPN software. In this paper we have studied the problem of combining multiple heterogeneous links into a single logical link to increase the throughput of individual TCP connections carried on a VPN. Our approach has been to marry two distinct contributions. First, we have used NATALIE, a traffic distributer that schedules packets to links dynamically. Link weights can be adjusted in real time via a companion application that measures changing network characteristics. We have also chosen to use a modified TCP congestion control algorithm to enhance the range of impaired links that can be successfully aggregated.

Proper selection of link weights is key to maximizing throughput over aggregated links. Indeed, we have shown that the sensitivity of throughput to link weights can be higher in aggressive transports schemes such as TCP-ER than in legacy TCP. But when chosen well, TCP-ER can significantly outperform legacy TCP, and its use can enable us to achieve aggregate gains on links that are sufficiently diverse that aggregation would yield no benefit with legacy TCP.

References

[1] Y. He, J. Brassil, "NATALIE: A Network-Aware Traffic Equalizer", *Proceedings of IEEE ICC*, June 2007.

[2] A. C. Snoeren, "Adaptive Inverse Multiplexing for Wide Area Wireless Networks," *IEEE GLOBECOM'99*, Rio de Janeiro, Brazil, December 1999, pp. 1665–1672.

[3] C. B. S. Traw, and J. M. Smith, "Striping Within the Network Subsystem," *IEEE Network*, vol. 9, no. 4, July/August 1995, pp. 22-32.

[4] H. Adiseshu, G. Parulkar, and G. Varghese, "A Reliable and Scalable Striping Protocol," Proceedings of ACM SIGCOMM'96, Stanford, CA, August 1996, pp. 131-141.

[5] J. Crowcroft, I. Phillips, TCP/IP and Linux Protocol Implementation, Wiley, New York, 2002.

[6] A. Bavier, et al, "Increasing TCP Throughput with an Enhanced Internet Control Plane," *Proceedings of Milcom'06*, Washington DC, 2006.

[7] Telecommunications Industry Association, "QoS Signalling for IP-QoS Support," *TIA-1039*, 2006.

[8] Emulab, http://www.emulab.net.

[9] I. Rhee, "BIC-TCP," http://www.csc.ncsu.edu/faculty/rhee/export/bitcp/.

[10] J. Li, J. Brassil, "On the Performance of Traffic Equalizers on Heterogeneous Communication Links," Proceedings of QShine'06, Waterloo CA, 2006.

[11] Iperf version 2.02, http://dast.nlanr.net/Projects/Iperf/.

[12] ITU Recommendation G.998 (G.BOND), http://www.itu.int/ITU-T/studygroups/com15/index.asp.

[13] Actelis Networks, http://www.actelis.com/.

[14] OpenVPN, http://openvpn.net/.

[15] H. Hsieh, R. Sivakumar, "Parallel Transport: A New Transport Layer Paradigm for Enabling Internet Quality of Service," *IEEE Communications Magazine*, vol. 43, no. 4, pp. 114-121, 2005.

[16] H. Hsieh, R. Sivakumar, "A Transport Layer Approach Achieving Aggregate Bandwidths on Multi-home Mobile Hosts," *ACM/Kluwer Mobile Networks and Applications Journal*, vol. 11, no. 1, pp. 99-114, 2005.

[17] D. S. Phatak and T. Goff, "A Novel Mechanism for Data Streaming Across Multiple IP Links for Improving Throughput and Reliability in Mobile Environments," *IEEE INFOCOM 2002*, New York, NY, June 2002.

[18] P. Sharma, SJ Lee, J. Brassil, K. Shin, "Distributed Channel Monitoring for Wireless Bandwidth Aggregation", *Proceedings of Networking 2004*, May 2004.

[19] J. C. R. Bennett, C. Partridge, and N. Shectman, "Packet Reordering is Not Pathological Network Behavior," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, December 1999, pp. 789-798.

[20] F. M. Chiussi, D. A. Khotimsky, and S. Krishnan, "Generalized Inverse Multiplexing for Switched ATM Connections," *Proceedings of IEEE GLOBECOM'98*, Sydney, Australia, November 1998, pp. 3134-3140.

[21] E. Blanton and M. Allman, "On Making TCP More Robust to Packet Reordering," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 1, January 2002, pp. 20-30.

[22] J. Duncanson, "Inverse Multiplexing," *IEEE Communications Magazine*, vol. 32, no. 4, April 1994, pp. 32-41.

[23] T. Jessup, "Emerging Technology: Balancing Act: Designing Multi-Link WAN Services," *IEEE Network Magazine*, June 2000.

[24] H. Hsieh, R. Sivakumar, "pTCP: An End-to-End Transport Layer Protocol for Striped Connections," *ICNP 2002*, pp. 24-33.

[25] M. Zhang, J. Lai, A. Krishnamurthy, L. Peterson, R. Wang, "A Transport Layer Approach for Improving End-to-End Performance and Robustness Using Redundant Paths", *Proceedings of USENIX'04*, June 2004.

[26] ATM Forum, "Inverse Multiplexing for ATM specification, Version 1.0", 1997.

[27] P. H. Fredette, "The Past, Present, and Future of Inverse Multiplexing", *IEEE_M_COM*, vol. 32, no. 4, pp. 42-46, 1994.