NATALIE: An Adaptive, Network-Aware Traffic Equalizer

Yihua He
University of California – Riverside
yhe@cs.ucr.edu

Jack Brassil
HP Laboratories
jack.brassil@hp.com

***Abstract*— The bandwidth of multiple physical communication links can be aggregated with inverse multiplexing to create a single, higher capacity logical communication link. However, achieving the maximum possible aggregated bandwidth becomes challenging as the communication characteristics of the underlying links (e.g., available bandwidth, packet loss rate) change with time.**

In this paper we introduce *NATALIE*, a Network-aware traffic equalizer. NATALIE combines an arbitrary set of network interfaces and schedules IP packet transmissions over those interfaces in a weighted round-robin fashion. To achieve the maximum possible 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. We describe NATALIE's implementation, the test environment we built on the *Emulab* Network Testbed, and the results of experiments demonstrating NATALIE's throughput performance for TCP connections over diverse underlying links with cross-traffic. [1]

I. Introduction

Inverse multiplexing is a conventional technique for aggregating the bandwidth of multiple communication links. In conventional settings, the underlying links used have been similar or identical (e.g., bonding of multiple T1s). But the increasing prevalence of multi-homed hosts and diverse access network technologies (e.g., DSL, cable) compel us to investigate the performance of aggregating the bandwidth of links with dramatically different communication characteristics. Consider, for example, a end system equipped with two wireless network interface cards of different maximum transmission rates (e.g., 802.11b, 802.11g) each connected to a separate access point. Each wireless link may be seeing different loss rates, and may be independently adapting its transmission rate in response to varying signal strength. How can these devices be aggregated to maximize TCP throughput?

In this paper we focus on the problem of partitioning a single TCP flow over multiple links. Previous research has shown that inverse multiplexing a TCP connection over heterogeneous links to obtain aggregated bandwidth is difficult. Splitting TCP segments belonging to a flow can and does result in out-of-order packet delivery that can result in a significant decrease in TCP throughput; sufficiently misordered packets trigger timeouts and unnecessary retransmission requests by the TCP receiver.

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

The TCP receiver 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 may mislead the sender into shrinking the congestion window unnecessarily, reducing throughput dramatically. TCP Selective Acknowledgements (*SACK*) allow 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 TCP congestion control mechanisms, TCP traffic is often unable to take advantage of bandwidth otherwise made accessible by multi-path networking. It is important to note, though, that many TCP variations exist, and each of these variants can and do perform differently in a multi-path setting. We focus on only one popular variant in this paper – TCP Westwood.

Naive attempts to aggregate two or more different links often yield very poor performance. Indeed, it is not uncommon to observe the throughput of 2 aggregated links approximately equaling the performance of the lowest quality link alone. Phillips & Crowcroft [11] report that round-robin scheduling becomes "unusable" on two links whose bandwidths differ by more than an order of magnitude. Of course, for heterogeneous links it is crucial to schedule packet transmissions to expedite delivery and minimize packet reordering, jitter, and load imbalance. Round-robin scheduling is typically limited to scheduling data over nearly homogeneous links, while weighted scheduling algorithms are more appropriate for heterogeneous links. Several research groups have shown that maximum throughput is achieved by assigning data to each channel in proportion to the channel's bandwidth-delay product [12], [17].

In this paper we present the design of NATALIE, a **N**etwork-**a**ware **tra**ffic equa**liz**e**r** with a packet scheduler that maximizes bandwidth aggregation by adapting to the available bandwidth of underlying links. Here we focus on the considerably more challenging problem of splitting a single TCP connection across multiple, heterogeneous links,

while constraining ourselves by making no change to TCP at either sender or receiver. Many researchers have shown that modifications of TCP can help maintain performance in multipath settings. The remainder of this document is organized as follows. Section II presents the design and implementation of the traffic equalizer, and the next section describes the testbed used to develop software and test its performance. Section IV describes the empirical performance of NATALIE when the characteristics of the underlying communication links are known, while Section V discusses system performance when dynamic cross-traffic is present. Section VI examines related work, and the final section summarizes our work.

II. NATALIE Implementation

NATALIE is based on a modifed version of a Linux-based traffic control kernel module called Traffic Equalizer (TEQL) that is included in the *iproute2* package and included in modern Linux distributions. TEQL assigns each incoming packet to one of *N* physical network interfaces using a deterministic round-robin discipline. Therefore, the traffic directed to each of the links is $1/N^{th}$ of the total traffic.

A strict round-robin discipline does not achieve optimal bandwidth aggregation when links have different communication characteristics (e.g., unequal transmission rates). To better utilize the aggregated bandwidth for disparate underlying links, Ji and Brassil [6] developed weighted-TEQL (wTEQL). In this kernel module, 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, or assigned in a fashion to achieve a desired load balance. However, the weights are assigned at module loading time and can not be changed until the module is unloaded and reinstalled in the system. This process takes at least tens of milliseconds, and disrupts active TCP connections. Further, wTEQL does not address the question of how to set link weights to maximize bandwidth aggregation, particularly as characteristics of the underlying links change due to factors such as changes in background traffic.

To address these limitations NATALIE permits dynamic, real-time adjustment of link weights on an installed module by user-level programs. NATALIE 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, preferably with root privileges, can dynamically adjust weights as network conditions change. In Section V we will show how NATALIE maintains high throughput as the controlling application measures the available bandwidth on each link, and adjust weights in proportion to those bandwidths.

III. Testbed

Emulab [1] 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,

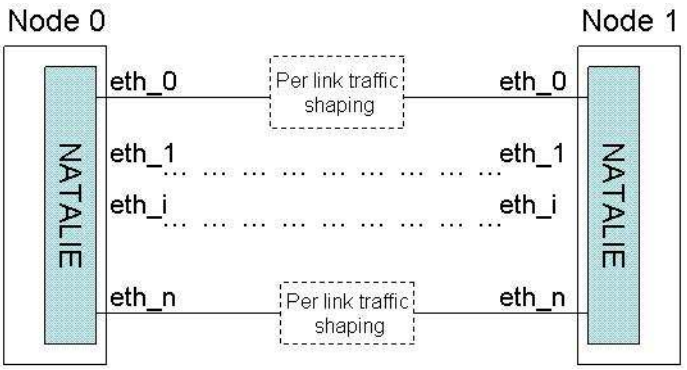


Fig. 1. A simple test topology on the Emulab testbed.

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 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 used the Distributed Internet Traffic Generator (D-ITG) [2] to generate background traffic on each link, as needed. We investigated three popular TCP versions available under the 2.6 kernel — BIC-TCP [3] (the default TCP choice since kernel version 2.6.8), TCP NewReno [5] and TCP Westwood [4]. Our experiments show that TCP-Westwood has better throughput than the other two in the presence of packet loss. This is due in part to the fact that TCP-Westwood does not regard a packet loss as an indication of congestion; instead, it counts the rate of returning ACKs to estimate bandwidth. Therefore, we choose to use TCP Westwood throughout this paper, and set the tcp_no_metrics_save option to ensure memoryless TCP operation from experiment to experiment. In all experiments TCP memory size was increased to ensure that sending window size was not limited by the system bandwidth-delay product.

IV. Performance Evaluation

In this section we present TCP throughput results for NATALIE operating on the topology of Figure 1. Emulated link parameters were fixed for each experiment, and no background traffic was present. We seek to answer questions such as the following: Suppose link A has relatively higher bandwidth than link B, but also suffers relatively higher packet loss. How should packets from a single connection be scheduled on the links to maximize the aggregated TCP throughput?

We next explore how NATALIE weights should be set to maximize throughput as the underlying link bandwidth, la-

tency and packet loss are changed. All presented throughput measurements were obtained using Iperf v2.02 [10].

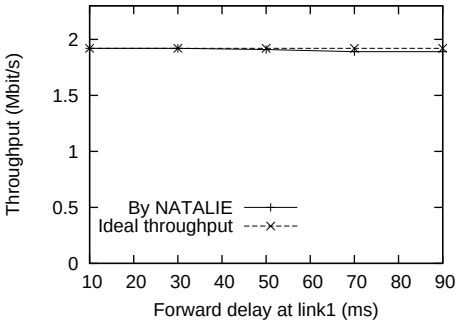
A. Two Links with differing delay

Packets from a single TCP connection traversing different links suffer different delays, both on the forward and reverse paths. In our experiments we transmitted data on the forward path, with ACKs returning on the reverse path. To distinguish the effects of delays on forward and reverse paths, we used 3 links (link0, link1 and link2) between 2 nodes with NATALIE forwarding data packets on link0 and link1 with equal weight, and *all* ACKs returning on link2. The delays on link0 and link2 were set at 10 ms., while the delay on link1 was varied. Here we take the *delay* to be the aggregate end-to-end delay including propagation, transmission and queuing delays. The delay range we considered (0-90ms) was set to cover the bulk of the end-to-end delays we would normally observe on either local or metropolitan area wireless networks, or intracontinental wired internet connections. Fig. 2 shows the TCP throughput measured for 120 seconds for each test on this aggregated link. For comparison, the graphs also show the maximum possible TCP throughput, which we take to be the sum of the maximum achievable throughput of each link when used separately.

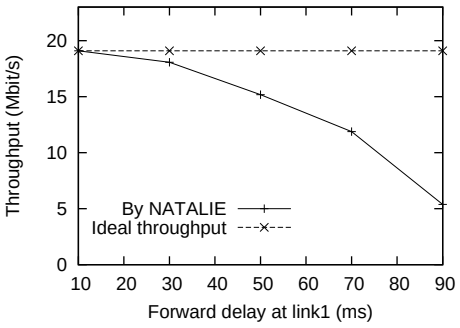
Fig. 2 (a) shows that for 1 Mbs links varying the delay difference on the forward path by changing the delay of link1 maintains high throughput over a very wide range of link delays. However, this was not the case when link bandwidth was increased to 10 Mbs; Fig. 2 (b) shows that the total throughput drops quickly when the latency on just one of the two forward paths increases.

Before examining the reasons for this throughput loss, we were curious whether a delay disparity on the reverse path would produce a similar result. So we reversed the direction of packet flow (i.e., forward on link0 and reverse on link 1 and link2) to examine the effect of ACK delay disparity. But Fig. 3 shows that that there is almost no throughput loss due to ACK delay disparity. This is likely the case because TCP ACKs are accumulative — even when the ACKs arrive out-of-order the TCP sender is still notified in a timely fashion.

Now we return to the falling throughput observed with delay disparity in the forward path on 10Mbits/s links. Figs. 4 (a)-(e) plot the throughput measured by iperf on the aggregated forward link at one second intervals for the first 120 seconds. For comparison we show the throughput for each individual link, and their sum. Figs. 4 (a)-(c) show that throughputs converge to the ideal within 120 seconds. However, convergence takes longer as forward delay disparity increases. Figs. 4 (d) and (e) show that the TCP throughputs fail to converge within the first 120 seconds. To see if it will eventually converge, we plot Fig. 4 (f), where measurement time is extended to 900 seconds. In the case where link1's forward delay is 70ms, the total throughput eventually converges to ideal after approximately 250 seconds. But when link1's forward delay reaches 90ms,

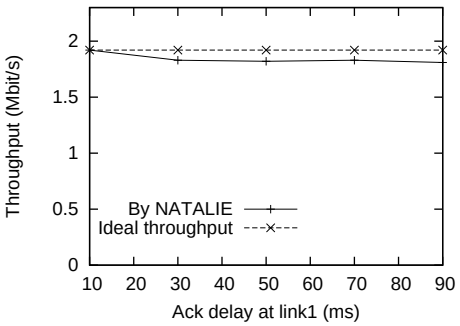


(a) 1Mbits/s links

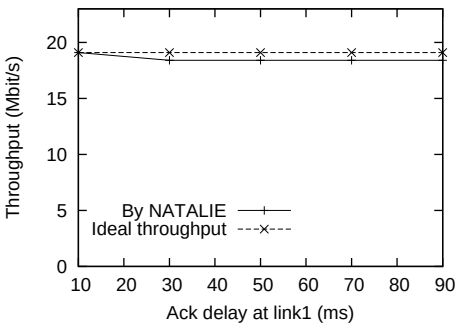


(b) 10Mbits/s links

Fig. 2. Throughput as delay disparity increases on the two links in the forward direction. The reverse path delay is fixed at 10 ms. for both links. The forward delay for link0 is fixed at 10 ms., and the delay of link1 is varied

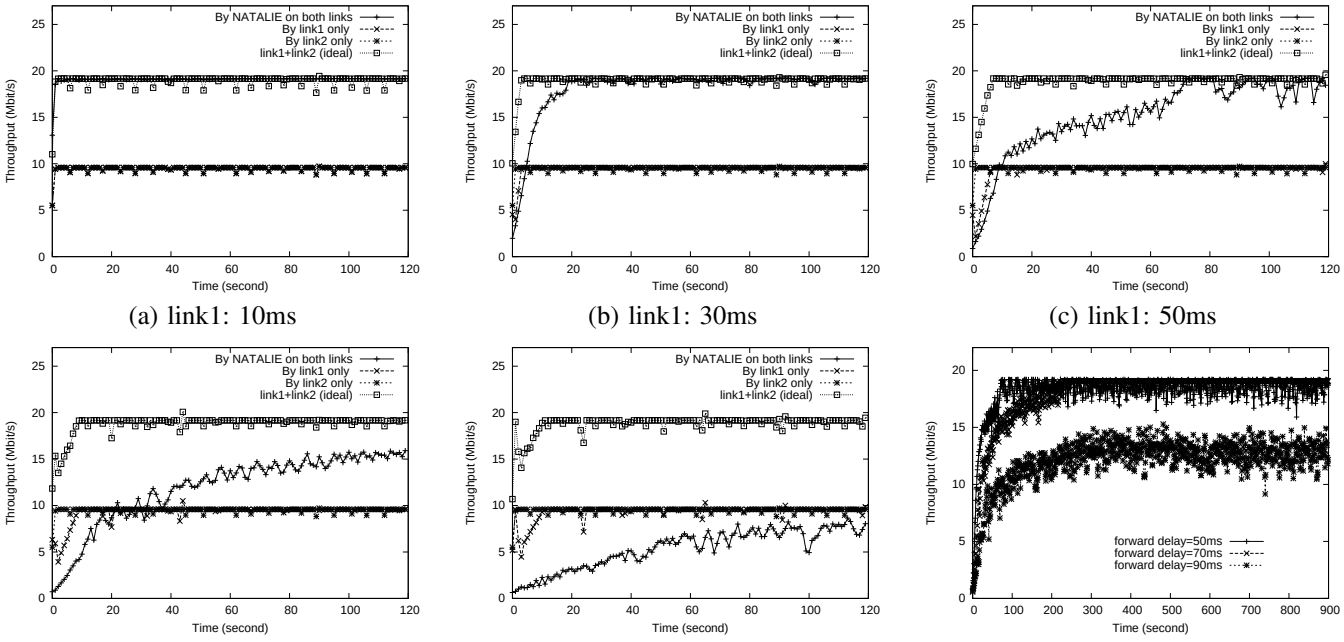


(a) 1Mbits/s links



(b) 10Mbits/s links

Fig. 3. Throughput as delay disparity increases on the two links in the reverse direction. The forward path delay is fixed at 10 ms. The reverse delay for link0 is fixed at 10ms., and the delay of link1 is varied.



(a) link1: 10ms (b) link1: 30ms (c) link1: 50ms

(d) link1: 70ms (e) link1: 90ms (f) large forward delay disparity

Fig. 4. Throughput convergence on links with increasing delay disparity.

the throughput does not converge to the optimal value and instead varies between 10-15 Mbits/s.

In summary, we find that delay disparity does not have a significant impact on TCP throughput, unless there is a huge delay disparity on relatively high bandwidth links. In practice, one-way delays between the same pair of nodes connected via multiple (even disjoint) paths rarely differ so dramatically.

B. Two links with differing loss

We next consider NATALIE's performance when aggregating lossy links with different packet loss rates (PLR). We focus on two questions. First, how much throughput gain, if any, can be achieved by aggregating multiple lossy links, compared to the throughput achievable using only the best component link? Second, how should we assign weights to links with differing PLR to realize the highest possible throughput?

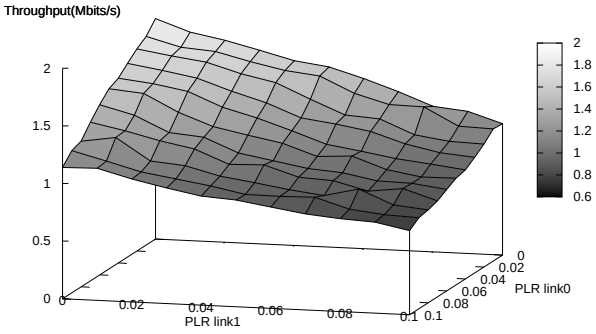
To answer the first question, we first look at the simple case of two heterogeneous links, and measure the overall throughput under various PLR combinations. For each PLR combination we empirically find the maximum throughput value by varying the weight ratio associated with the two links from 0 to ∞. Figs. 5 (a) and (c) show the resulting throughput for 1 Mbs and 10 Mbs links, respectively. For comparison we also plot the throughput achievable on the higher performing of the two links in Figs. 5 (b) and (d). For the case of 1 Mbs links we can see that combining two links with NATALIE can achieve much higher throughput than just using the better of the two links, in most cases. The total throughput drops when packet loss increases, only reaching the same throughput as the better performing component link when both links have PLR=0.1, a loss rate considered quite high in most operational network settings.

For 10 Mbs links NATALIE achieves large throughput gain when the PLR on each of the two links is small (≤ 0.01). When the PLR reaches values between 0.01 and 0.04, throughput on the aggregated link is only slightly better than only using the better link. When the PLR on both links exceeds 0.04, the throughput achieved by aggregating links is actually lower than just using the better one. In summary, our results indicates that high PLRs can significantly decrease NATALIE's throughput, even at the optimal weight ratio. Further, additional experiments show that the larger the bandwidth of each link, the larger throughput decrease a given PLR will produce. When the PLRs of both component links are large, it is invariably better to use only one link than to try aggregating two or more.

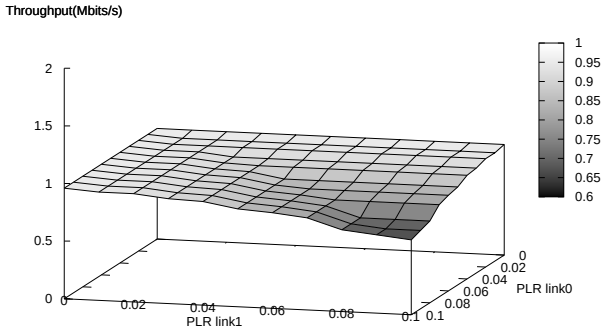
We next return to the second question we posed early in this section, namely how we can maximize throughput by assigning weights to links of different bandwidth and different packet loss rates. To address this question we examine three cases:

- Case I: One lossless link and one lossy link with identical bandwidth.
- Case II: One lossless link and one lossy link with lower bandwidth.
- Case III: One lossless link and one lossy link with higher bandwidth.

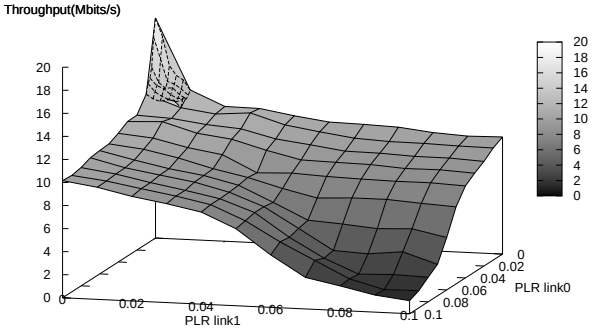
In each case we consider NATALIE operating on two links (link0 and link1) where the weight of link0 is set to $w_0 = 100$; that is, if the weight of link1 is $w_1 = 50$, then $(50 +$



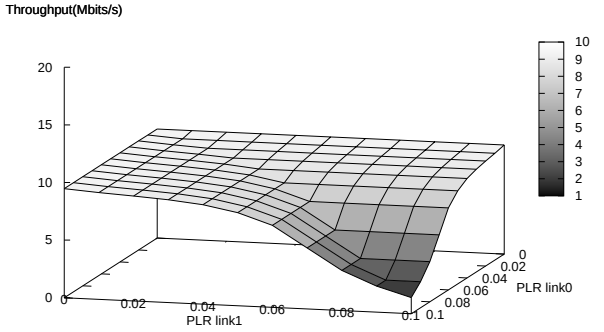
(a) Throughput with NATALIE on two 1 Mbs links



(b) Throughput of the better performing 1 Mbs link only



(c) Throughput with NATALIE on two 10 Mbs links



(d) Throughput of the better performing 10 Mbs link only

Fig. 5. The maximum achievable throughput for two aggregated links and the best performing component link under all packet loss rate combinations.

$100)/150$ or $1/3$rd of the packets are assigned to link1. In addition, to simplify exposition we fixed the PLR of link0 to be zero ($plr_0 = 0$) in all experiments.

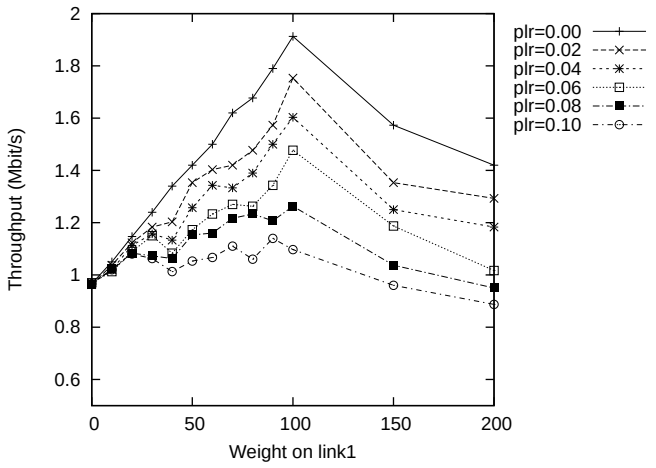
These configurations are deceptively complicated, and intuition provides little sure guidance on how to assign link weights to maximize throughput. For example, a plausible hypothesis might be to decide to weight links in proportion to the maximum achievable bandwidth of each component. However, as we will see shortly, this conjecture is not generally correct.

Case I: Figs. 6 (a) and (b) show the throughput of two links with identical bandwidth. For 1 Mbs links (Fig. 6 (a)) observe that throughput is maximized when the weight of link1 equals that of link0 ($w_1 = w_0 = 100$). In general, our experiments show that the maximum is achieved when the weight ratio equals the ratio of the respective link bandwidths (b_1/b_0) as long as the PLR of the lossy link remains modest ($0 \leq plr_1 \leq 0.1$).

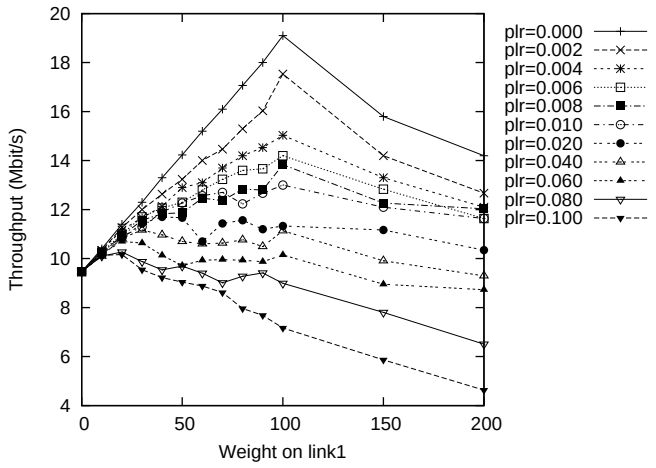
This result also holds when the PLR is small ($plr_1 \leq 0.01$) for 10 Mbits/s links (Fig. 6 (b)). However, as the PLR for one of the 10 Mbs links increases, the weight ratio (w_1/w_0) that maximizes throughput decreases. Note that when the highest throughput is achieved by assigning a weight ratio $w_1/w_0 \neq 1$, the throughput is only marginally higher than the throughput achieved using only the better performing component link.

Case II: Figs. 6 (c) and (d) show NATALIE's throughput in the case where a lossless high bandwidth link pairs with a lossy low bandwidth link. As in Case I, the throughput peaks at $w_1/w_0 = b_1/b_0$ when the plr_1 is small. As the PLR increases the weight ratio maximizing throughput becomes less than b_1/b_0. However, in these cases, the throughput achieved by assigning weights according to the ratio b_1/b_0 is only marginally less than the highest achievable throughput. The only exception is when the PLR of the low bandwidth link is very high ($plr_1 \geq 0.1$). However, in such a situation it is prudent to use only the lossless high bandwidth link, because this will achieve small throughput degradation, while realizing a shorter convergence time to maximum throughput.

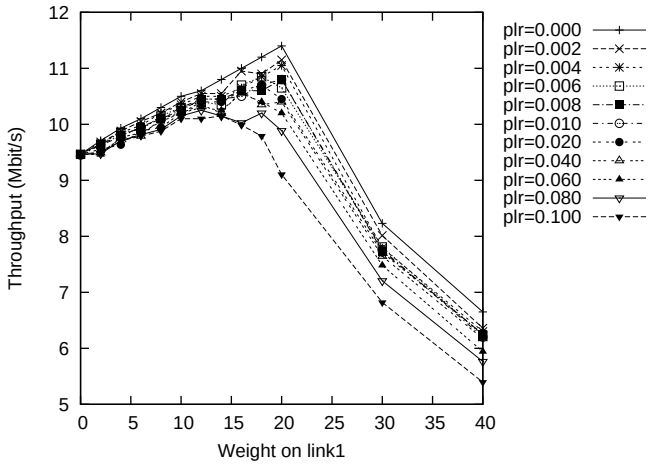
Case III: Figs. 6 (e) and (f) show the throughput of two



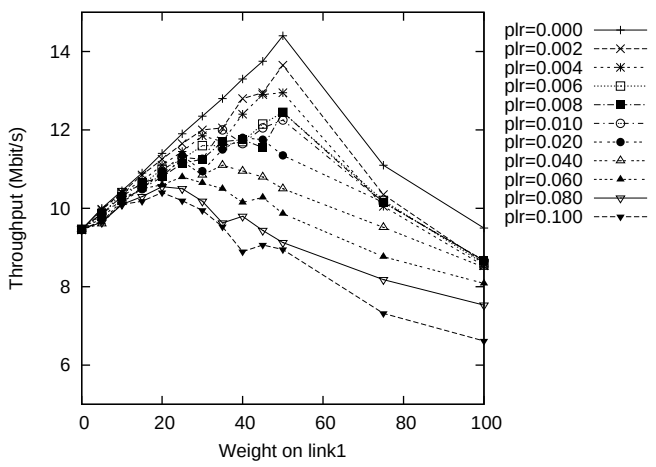
(a) $b_0 = 1$ Mbs, $b_1 = 1$ Mbs



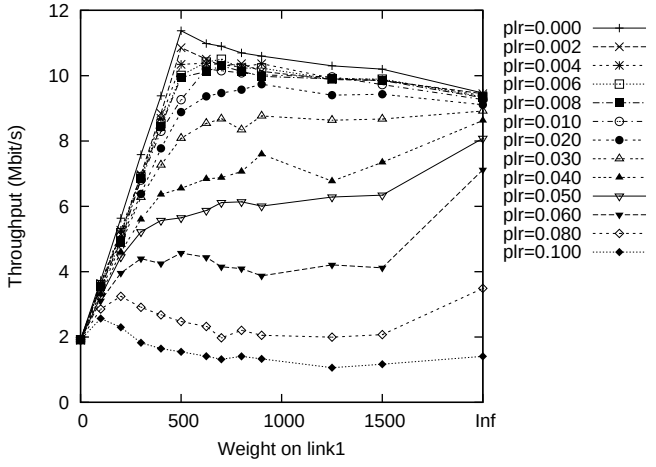
(b) $b_0 = 10$ Mbs, $b_1 = 10$ Mbs



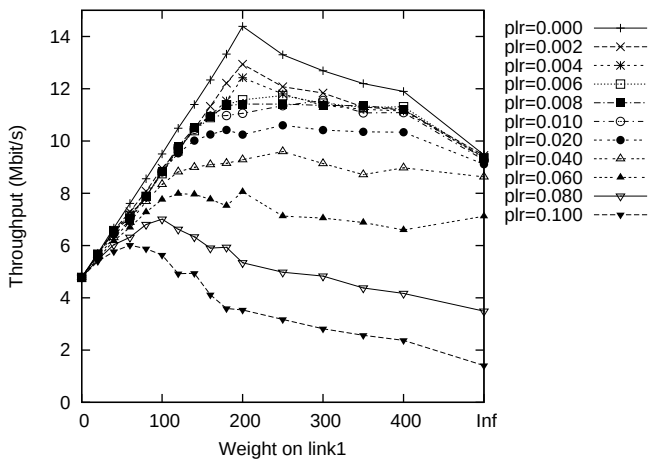
(c) $b_0 = 10$ Mbs, $b_1 = 2$ Mbs



(d) $b_0 = 10$ Mbs, $b_1 = 5$ Mbs



(e) $b_0 = 2$ Mbs, $b_1 = 10$ Mbs



(f) $b_0 = 5$ Mbs, $b_1 = 10$ Mbs

Fig. 6. Throughput of NATALIE as the weight of link1 is adjusted to compensate for differing bandwidth and PLR on the component links. The PLR of link0 is set to 0.0, and the weight assigned to link0 is fixed at 100.

links where a lossless low bandwidth link (either 2 or 5 Mbs) pairs with a lossy high bandwidth link (10 Mbs). Once again we find that, when the PLR is small, assigning $w_1/w_0 = b_1/b_0$ achieves the highest throughput. However, when PLR increases the highest throughput is typically achieved by just using the lossy, high bandwidth link alone. When PLR grows larger, the highest throughput is achieved by assigning a weight ratio w_1/w_0 to a value in the interval $0 \leq w_1/w_0 \leq b_1/b_0$.

These results cause us to propose a simple guideline for assigning link weights when using NATALIE to schedule packets associated with a TCP connection to component links with no cross traffic. This guideline is not a precise specification of an optimal assignment, but rather a rough rule-of-thumb likely to realize good throughput performance. This rule can be summarized as follows: 1) In the absence of a large disparity in packet loss rate or delay between the component links, the link weight assignments should be proportional to the component links' bandwidths. 2) As the disparity of the link impairments on the component links increases, overweight the highest bandwidth link. 3) As the disparity of the link impairments on the component links grows large, avoid using multiple links entirely by assigning all packets to the highest bandwidth component link.

V. Dynamic Auto-weighting in the Presence of Cross-Traffic

In the previous section we explored the throughput of NATALIE on heterogeneous component links. To isolate the effects of component link impairments, we excluded the possibility of cross-traffic on those links. In this section we explore how we have designed NATALIE to maximize throughput by adapting to the presence of background traffic.

NATALIE permits authorized user programs to adjust its link weights. In a typical application, an external user program can probe component link communication characteristics such as available bandwidth, packet loss rate, and round-trip latency, and assign the weights to maximize throughput.

To test the operation of this mechanism, we constructed a simple bandwidth estimation program for directly connected links on an end system. The program is either informed of the transmission rate of locally attached interfaces, or obtains this information from a standard Linux networking utility such as ethtool. The program counts the number of bytes transferred on each link in a specified time interval, and calculates the available bandwidth on each link. These measurements are then combined and filtered to calculate a link weight set, and the resulting values are written to the /sys/module/NATALIE/parameter/weights directory to set current link weights. We call this operation "auto-weighting."

Fig. 7 depicts a NATALIE auto-weighting experiment, and compares the results to the performance obtained with the native TEQL module, which assigns packets in a strict round-robin to component links. The background traffic generated by D-ITG on two 10 Mbs links is shown in Fig. 7 (a). This traffic starts at time $t = 10$ seconds and ends at $t = 110$ seconds, with the traffic pattern changing every 20 seconds in that period. Fig. 7 (b) shows achievable system throughput using the TEQL module to assign traffic equally to each of the two component links independent of the level of background traffic. The dotted line depicts the maximum achievable bandwidth – the sum of the available bandwidth on each of these two links. The figure clearly shows that the total throughput achieved by TEQL is frequently not close to the ideal throughput. On the other hand, Fig. 7 (c) shows the throughput achieved using NATALIE with auto-weighting. The throughput achieved is very near the the maximum available bandwidth; occasional sharp downward spikes correspond to events where background traffic changes, and the system requires several measurement intervals to recalculate available bandwidth and adjust link weights accordingly.

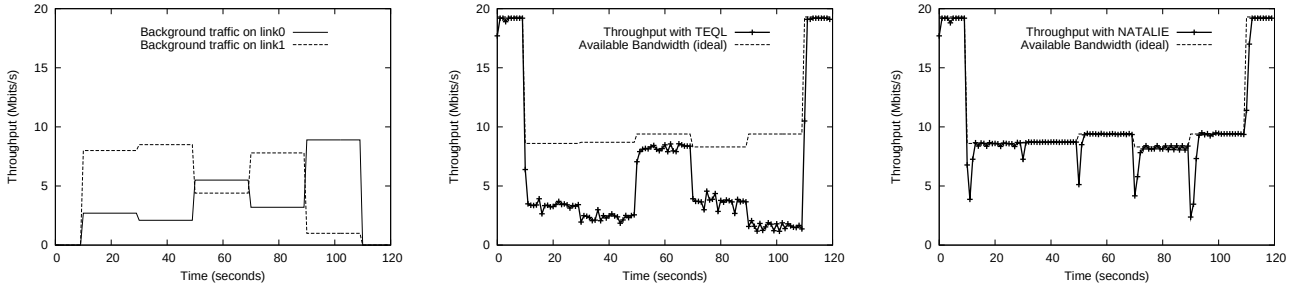
VI. Related Work

A large number of researchers have investigated various techniques to aggregate bandwidth at every possible protocol layer. Though pursuing similar objectives, these schemes differ dramatically 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. A recent example of this approach is the hardware based *ethernet over copper* bonding product from Actelis Networks [14] that achieves symmetric access up to 70 Mbs over multiple DSL wire pairs using ITU Recommendation G.998 (G.Bond) [13].

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, these 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 [18] [17], [19].

Handling diverse links has focused attention on approaches to modifying TCP to support transmission over multiple channels [15], [16]. 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 in this approach is the requirement



(a) Background traffic (b) Throughput with TEQL (c) Throughput with NATALIE

Fig. 7. Auto-weighting significantly improves throughput.

to modify the end system TCP stacks for *all* participating end systems. Though modifying TCP can be highly effective at achieving aggregation gain, it unfortunately requires new kernels to be deployed on all participating end systems.

By contrast, the approach we explore in this paper uses loadable kernel modules that can be added or removed from participating systems on demand, without requiring kernel modifications. Further, NATALIE can be installed on a sender alone to provide aggregation, with no need for any receiving systems to be similarly configured.

VII. Conclusion

As multi-homing becomes increasingly prevalent and end systems have access to multiple communications networks, we are faced with the challenge of using these communication links effectively while not requiring change to either communication infrastructure or widely deployed protocols such as TCP. In this paper we have studied the problem of combining multiple heterogeneous links into a single logical link to increase the throughput of a TCP connection. We have made two distinct contributions. First, we have developed a Linux kernel module called NATALIE, a traffic distributer that schedules packets to links dynamically. The weighting can be adjusted in real time, without unloading the module, and supports network-awareness via a companion application that measures network characteristics.

Second, we have extensively examined the throughput performance for NATALIE on multiple heterogeneous links under various combinations of delay, packet loss rate and bandwidth. We have found that, to achieve the best TCP throughput under a wide range of network conditions – including those commonly found in operational wired and wireless networks – it is normally 'best' to set the ratio of the weights on the component links in proportion to the ratio of their bandwidths. Further, we have identified conditions under which it is normally better to use the best performing component link alone rather than attempting to aggregate links.

In our future research we will seek to expand the network settings where NATALIE can be used. In particular, we see NATALIE scheduling packets on disjoint paths that can extend many hops into the network. The challenge here is to discover the characteristics of bottleneck links located deeper in the network. We envision incorporating existing, end-to-end available bandwidth estimation tools (e.g., pathChirp[7], pathLoad[8], Spruce[9]) to probe network paths and provide measurements needed for auto-tuning.

References

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

[2] D-ITG, http://www.grid.unina.it/software/ITG/.

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

[4] Ren Wang, Giovanni Pau, Kenshin Yamada, M. Y. Sanadidi, Mario Gerla, "TCP Startup Performance in Large Bandwidth Delay Networks," Proc. of INFOCOM 2004, Hong Kong, March 2004

[5] S. Floyd, T. Henderson, "The New-Reno Modification to TCP's Fast Recovery Algorithm," RFC 2582, April 1999.

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

[7] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil and L. Cottrell, "pathChirp: Efficient Available Bandwidth Estimation for Network Paths," Passive and Active Measurement Workshop, 2003.

[8] M. Jain, C. Dovrolis, "End-to-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput," ACM/IEEE Transactions on Networking, 2003.

[9] J. Strauss, D. Katabi and F. Kaashoek, "A Measurement Study of Available Bandwidth Estimation Tools," The Internet Measurements Conference, Florida, 2003.

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

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

[12] 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.

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

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

[15] L. Magalhaes and R. Kravets, "MMTP: Multimedia Multiplexing Transport Protocol," *Proceedings of the first ACM Workshop on Data Communications in Latin America and the Caribbean*, San Jose, Costa Rica, April 2001, pp. 220-243.

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

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

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