Overlay TCP: Ending End-to-End Transport for Higher Throughput

Himabindu Pucha and Y. Charlie Hu
Purdue Univeristy
West Lafayette, IN 47907
{hpucha, ychu}@purdue.edu

ABSTRACT

Recently, overlay routing has emerged as a promising approach to mitigating many problems with Internet routing, such as improving the reliability of Internet paths as well as the throughput via multiple paths. We advocate that as overlay routing is gaining wider acceptance, it is time to investigate TCP performance on top of overlay routing, in particular, the potential of overlay routing for improving the end-to-end TCP throughput. We propose a general-purpose application-level protocol extension to TCP, called Overlay TCP (oTCP), that improves TCP throughput by splitting a TCP connection with a large RTT into pipelined TCP sub-connections with small RTTs via overlay routing.

1. INTRODUCTION

TCP has been the primary transport protocol used in the Internet for over 20 years. Improving TCP throughput is of importance due to the wide variety of applications that use TCP. A performance characteristic of TCP is that the rate at which its window size increases is inversely proportional to the average round trip time (RTT), and the average window size is directly related to the throughput. Thus, reducing the RTT between connection endpoints can increase the window size faster, resulting in increased throughput. An intuitive way to reduce the effective RTT of a direct TCP connection between any two hosts is to split that connection into multiple pipelined sub-connections such that each sub-connection has a lower RTT than the direct connection.

We observe that the emergence of overlay networks provides a natural platform that can be leveraged to support splitting TCP connections by allowing peer nodes to select each other as intermediaries. Based on this observation, we propose Overlay TCP (oTCP), an application-level protocol extension to TCP that splits an end-to-end TCP connection with a high RTT into pipelined sub-connections with low RTTs using intermediate nodes to achieve higher end-to-end throughput. In addition to increasing throughput, oTCP can also route around failures and discover paths that are better than the direct path. Thus, ending end-to-end transport has the potential to not only improve the throughput of a connection but also improve the reliability and quality of paths between endpoints. We advocate oTCP as a viable architecture for throughput demanding applications.

We argue that oTCP is more desirable than previous approaches towards increasing TCP throughput such as changing congestion control, changing the initial window size, or using parallel sockets for two reasons. First, oTCP builds on top of unmodified TCP. Second, on any single Internet path, an oTCP flow is not unfair to other flows, i.e., the TCP protocol behavior is unchanged. oTCP is also fundamentally different from other end-to-end overlay-based schemes such as Detour [6] and RON [1] since it terminates overlay connections at the transport rather than the network layer to harness the reduction in RTT; it can improve the TCP throughput even if the sum of sub-path RTTs is not smaller than the end-to-end RTT.

Table 1: TCP throughput gain between nodes at NCSU and Caltech.

Hop count	1 (Direct)	2	3	4
Path RTT (msec)	75.7	63.5	47.1	39.3
Throughput (Kbps)	3097.1	3505.4	4695.0	5391.0

2. THE THROUGHPUT POTENTIAL OF OTCP

oTCP splits the direct TCP connection between any two hosts (say A and B) into a multi-overlay-hop path (called multi-hop paths henceforth), where *each overlay hop is an independent TCP connection*, such that the RTT of each overlay hop is lower than the direct RTT between A and B. We denote the maximum of all the overlay hop RTTs as the *path RTT*, i.e., $path\ RTT = max(RTT_i),\ \forall i \in [1, n]$, where RTT_i is the RTT of the ith hop. In this scenario, each hop's throughput will be governed by that hop's RTT and thus will be individually higher than the direct throughput between A and B. Further, once a TCP transfer starts, the data bytes will be pipelined through the multi-hop path, and the end-to-end throughput between A and B is constrained by the throughput of the worst overlay hop, i.e., the one with the highest RTT.

To assess the potential throughput benefit of oTCP in a wide area network, we measured the end-to-end throughput improvement from splitting a TCP connection transferring a 4MB file using nodes from an example overlay network, PlanetLab [4]. The results, depicted in Table 1, show that the TCP throughput improves from 3097 Kbps to 5391 Kbps by splitting the direct TCP connection between NCSU and Caltech into 4 hops using 3 intermediate nodes (U. Tenn., WUSTL, and U. Texas).

We also performed an assessment of the potential throughput benefits of oTCP in the Internet via a large scale measurement study using 313 PlanetLab nodes. The strong cor-

ACM SIGCOMM 2005, August 22–26, 2005, Philadelphia, PA.

Table 2: oTCP potential based on p2p RTT trace.

Nodes in trace	313
Total number of connections	88841
Number of connections without benefit from oTCP	622
Number of connections with benefit from oTCP	88219
Average reduction in *path RTT* (%)	56.2%

relation between RTT and TCP throughput as observed in Table 1 allows us to restrict our analysis here solely based on RTTs. We measured the RTTs between these nodes to form our *p2p RTT trace*, as well as from these nodes to a list of 1,679 popular web servers to form our *web RTT trace*. The measurements were then analyzed to identify the number of cases in which the RTT between a pair of nodes is reduced when using oTCP and the amount of reduction in RTT in each such case. Table 2 shows that in the p2p RTT trace, out of the 88,841 connections, 88,219 have lower RTT paths that consist of one or more intermediaries. Thus, oTCP can potentially benefit 99.2% of the connections in this network. Table 2 also demonstrates the extent of benefit possible using oTCP; the RTT is on an average reduced by 56% (a factor of 2.3) for a transfer. Similar results are obtained for the web RTT trace. These results show that oTCP has the potential to bring substantial throughput improvement to the standard TCP. We also found that similar to other overlay routing systems, oTCP was also able to repair 99% of the broken direct paths.

3. DESIGN AND EXPERIMENTS

Although oTCP is conceptually simple, its architectural design requires answers to many fundamental questions: (1) How many hops are required to obtain reasonable benefit using oTCP? (2) How should the intermediaries be chosen without incurring high overhead and delay? (3) What transfer sizes can benefit from oTCP? To address these issues, we performed analysis using the web and p2p RTT trace and measurements using the PlanetLab nodes. The details of the analysis can be found in [3].

Our analysis using the RTT traces from PlanetLab shows that most of the benefit can be obtained by splitting the TCP connections using 1 or 2 intermediaries, i.e., 2 or 3 hops, and using additional intermediaries provide diminishing returns. Taking the delay and overhead penalty required to obtain a 3-hop path into consideration, we conclude that 2-hop paths provide a cost-effective means of splitting TCP for reduced path RTT and consequently increased TCP throughput.

We also studied different strategies to pick the single intermediary (overlay) node that splits the RTT between two endpoints. We found that a random selection strategy does not work well due to the large sampling size needed. We then designed a highly effective geographic clustering algorithm that groups the overlay nodes and picks one overlay node from each cluster.

Finally, we observed the benefit from TCP splitting is apparent even in medium-sized file transfers. We have developed a prototype implementation of oTCP using Netfilter to transparently hijack TCP connections and route them through the intermediaries. Using a web-browsing workload of 100 randomly selected files which was driven repeatedly using a `wget` client on a daily basis over a period of one week, oTCP was found to improve the TCP throughput for 85% of the file transfers. In particular, 50% of the transfers had a throughput increase of more than 25%. Furthermore, oTCP never performed worse than the direct transfer during our evaluation period. Although 10-15% of the transfers could not get a large benefit from oTCP, we found that all of these transfers were to web servers situated close to the client in which case the direct path is likely to be optimal.

4. OPEN QUESTIONS

The deployment and use of oTCP has several implications. oTCP blurs the end-to-end nature of TCP since an ACK now signals data reception at an intermediary. Since oTCP preserves the semantics of `close` and `connect` by replaying all connection terminations and server failures that occur at one end of a transfer to the other end, we expect most applications would be unaffected from using oTCP since oTCP ensures that the status of one end of the connection is replicated at the other end. An interesting question remains as how to extend oTCP so that it has both end-to-end reliability and per-hop congestion control.

Using oTCP also has implications to TCP fairness. Although each overlay hop of oTCP uses unmodified TCP, the end-to-end route selection in oTCP is selfish [5] in nature as it allows end users to greedily select routes to optimize their own performance without considering the network-wide criteria. However, in contrast to multi-path TCP, oTCP's impact is more predictable since each sub-connection behaves like a normal TCP connection sharing resources fairly with other flows.

Wide deployment of oTCP has other implications. First, as more clients are using oTCP and thus sharing intermediaries, the intermediaries can potentially become congested and incur queuing delay. Second, in addition to RTT, packet loss can potentially affect the relative throughput of different TCP connections. When this occurs, formula-based throughput prediction [2] can be used in picking intermediaries. Third, since oTCP is primarily designed to improve TCP throughput, it is more likely to be adopted for bandwidth-intensive applications such as FTP, HTTP, and p2p file transfers for which TCP packets can be hijacked based on port numbers. oTCP can also be used as a building block in other bandwidth-intensive distributed systems such as application-layer overlay multicast, grid computing applications, and parallel (multi-path) TCP.

5. REFERENCES

[1] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient Overlay Networks. In *Proc. of ACM SOSP*, 2001.

[2] Q. He, C. Dovrolis, and M. Ammar. On the predictability of large transfer TCP throughput. In *Proc. of ACM SIGCOMM*, 2005.

[3] H. Pucha and Y. C. Hu. Overlay TCP: Multi-Hop Overlay Transport for High Throughput Transfers in the Internet. Technical Report TR-ECE-05-08, Purdue University, 2005.

[4] PlanetLab. http://www.planet-lab.org.

[5] L. Qiu, Y. R. Yanga, Y. Zhang, and S. Shenker. On selfish routing in internet-like environments. In *Proc. of ACM SIGCOMM*, 2003.

[6] 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*, 19:50–59, 1999.