Handling Packet Losses in Cloud-based MPTCP Application Traffic

Kiran Yedugundla, Per Hurtig and Anna Brunstrom
Dept. of Computer Science, Karlstad University, Karlstad, Sweden

Keywords: Multipath TCP, Cloud Applications, Latency, Loss Recovery, Performance Evaluation, Measurements.

Abstract: Internet traffic is comprised of data flows from various applications with unique traffic characteristics. For many cloud applications, end-to-end latency is a primary factor affecting the perceived user experience. As packet losses cause delays in the communication they impact user experience, making efficient handling of packet losses an important function of transport layer protocols. Multipath TCP (MPTCP) is a modification to TCP that enables simultaneous use of several paths for a TCP flow. MPTCP is known to improve throughput. However, the performance of MPTCP is not optimal when handling certain loss scenarios. Efficient packet loss recovery is thus important to achieve desirable flow completion times for interactive cloud-based applications. In this paper we evaluate the performance of MPTCP in handling tail losses using traffic traces from various cloud-based applications. Tail losses, losses that occur at the end of a flow or traffic burst, are particularly challenging from a latency perspective as they are difficult to detect and recover in a timely manner. Tail losses in TCP are handled by using a tail loss probe (TLP) mechanism which was adapted to MPTCP from TCP. We investigate the performance of TLP in MPTCP, comparing the standard implementation to a recently proposed, less conservative approach. Our experimental results show that a less conservative implementation of TLP performs significantly better than the standard implementation in handling tail losses, reducing the average burst completion time of cloud based applications when tail loss occurs by up to 50% in certain cases.

1 INTRODUCTION

Cloud applications contribute to a significant amount of the Internet traffic. The interactive nature of many cloud-based applications such as Google Docs and Bing Maps make them sensitive to latency. User experience is significantly affected in such applications when data delivery is delayed. Data delivery between end-hosts is often carried out by reliable transport layer protocols such as Transmission Control Protocol (TCP) (Postel, 1981). The evolution of transport layer protocols in pursuit of performance improvement, led to an extension of standard TCP that enables data transfer using multiple flows known as Multipath TCP (MPTCP) (Ford et al., 2013). MPTCP improves end-to-end throughput of connections with simultaneous use of multiple paths between end points. However, application perceived latency is often affected more by losses in transmission than throughput. Thus, a transport protocol that can offer better loss recovery and throughput is an optimal choice for interactive cloud-based applications.

Applications perceive multiple MPTCP subflows as a single TCP flow. To ensure transparency, MPTCP handles several functions such as scheduling, packet reordering, and loss recovery. This paper focuses on handling losses in a tail loss scenario, that is identified as a major cause of higher delays in short flows (Dukkipati et al., 2013), especially due to the way in which the lost packets are recovered. Tail losses are packet losses that occur at the end of a flow or a traffic burst. TCP handles tail losses using a technique known as tail loss probe (TLP) (Dukkipati et al., 2013). MPTCP applies TLP for each subflow independently in its implementation and takes a conservative approach when it comes to loss recovery and avoids using multiple subflows for recovery probes. A recent approach in (Yedugundla et al., 2017) reduces latency by improving the loss recovery in the event of tail loss using MPTCP, and illustrates benefits in certain cases using synthetic traffic.

Results achieved with synthetic traffic provide a trend in an ideal network condition and does not consider the real world adverse affects of network technologies. In this paper, we evaluate the latency performance of both approaches using real world cloud based application traffic. We consider a set of traffic scenarios from Google Maps, Google Docs and Netflix to cover a range of different traffic patterns in cloud-based applications. Our experiments sug-

Yedugundla, K., Hurtig, P. and Brunstrom, A.
Handling Packet Losses in Cloud-based MPTCP Application Traffic.
DOI: 10.5220/0007723801110119
In *Proceedings of the 9th International Conference on Cloud Computing and Services Science (CLOSER 2019)*, pages 111-119
ISBN: 978-989-758-365-0

gest that the less conservative approach proposed in (Yedugundla et al., 2017) can provide a significant latency reduction in the considered path failure scenarios.

The rest of the paper is organized as follows. In Section 2, we discuss the background to loss recovery in TCP. The related research on improving latency with loss recovery is available in Section 3. Section 4, provides a detailed explanation of tail loss handling in MPTCP. In Section 5, we discuss the MPTCP experiment setup and the traffic scenarios used in the experiments. Section 6, provides results on the latency performance of the MPTCP TLP variants that are considered for evaluation. We conclude the paper in Section 7 with a summary of the results and future research directions.

2 BACKGROUND

Flow completion time is one of the metrics that determines the quality of an interactive flow or a latency sensitive application. Retransmission schemes affect the flow completion time. TCP recovers lost packets by retransmitting the same. If an acknowledgement (ACK) for a sent packet is not received in a certain amount of time, a retransmission timeout (RTO) occurs and the packet is resent. Fast retransmit uses duplicate ACKs to detect packet loss faster and retransmits a previously sent and unacknowledged packet once a certain number (i.e., three) of duplicate ACKs have been received. TCP has to rely on an RTO if the number of duplicate ACKs are insufficient to trigger a retransmission. On an RTO event, TCP retransmits all unacknowledged packets. The RTO value is set rather conservatively, usually several times of the round-trip time (RTT) and subsequently updated based on the RTT. Limited Transmit (Allman et al., 2001), SACK-based fast recovery (Blanton et al., 2012), and Early Retransmit (Allman et al., 2010) have improved the retransmission strategies for TCP. These algorithms help triggering fast retransmit when the congestion window is small and insufficient to create triple duplicate ACKs.

When there is enough data to transmit from the sender, the packets sent after the lost packet trigger duplicate ACKs. However, if the loss occurs at the end of a packet flow, TCP has to wait for an RTO to retransmit the lost packets. Tail Loss Probe (TLP) is a loss recovery mechanism proposed in (Flach et al., 2013), to recover tail losses using a probe mechanism instead of waiting for an RTO event. A probe is normally the last transmitted packet or an unsent packet, with a shorter timeout value than the RTO, called as probe timeout (PTO). The value of PTO is computed to be more than the RTT and less than the RTO to avoid spurious retransmissions, which is an issue with smaller RTO values. Moreover, an RTO reduces the congestion window and triggers a slow-start unlike PTO. TLP provides significant improvement in tail loss scenarios and is available in the Linux TCP implementation.

In TCP, the retransmission mechanisms use sequence numbers to identify lost packets and handle retransmissions. MPTCP has two levels of sequence numbers to support efficient data transfer, namely data sequence numbers and TCP sequence numbers. Data sequence numbers are for the end-to-end data transfer and TCP sequence numbers are for an individual flow data sequence. Within a subflow, there is an association between data sequence numbers and TCP sequence numbers. A loss occurring in an individual TCP flow corresponds to a loss in end-to-end data. Multipath TCP, as an extension of TCP, uses the TCP retransmission mechanism. However, the interaction between the two levels makes the problem of retransmitting more challenging than in TCP. The dual sequence numbering enables MPTCP to respond to loss of packets by retransmitting the lost packets on an alternate path. The Linux implementation of MPTCP uses a set of retransmission heuristics to handle retransmissions. In addition to the normal TCP retransmission on the subflow level, the data outstanding on a timed-out subflow is rescheduled for transmission on a different subflow using timeout as the indicator. Fast retransmit on a subflow does not trigger retransmission on another subflow.

3 RELATED WORK

Prior research on MPTCP retransmission techniques can be classified as conservative or redundant. Use cases of MPTCP, such as simultaneous use of WLAN and 4G, in general involve a degree of asymmetry, leading to subflows with different delay characteristics. Experimental studies such as (Yedugundla et al., 2016) investigate the latency performance of MPTCP using various traffic types and path asymmetry. In cases with significant delay differences between paths, opportunistic retransmission (Raiciu et al., 2012) improves the latency by using the fastest path to retransmit the data originally sent on another path. Authors of (Chen et al., 2016), provide a mechanism that exploits the path diversity by quickly retransmitting on the fastest paths. Such quick retransmission comes with a cost of redundant packets as each individual TCP flow should retransmit always as well to follow TCP semantics. Lower latency can be achieved with full redundancy on all MPTCP paths (Frommgen et al., 2016) at a cost of network overhead to support interactive applications. For bursty traffic, scheduling packets based on loss and delay can improve latency (Dong et al., 2017). The MPTCP standard (Ford et al., 2013) suggests a more conservative approach, that retransmits the lost packets on another path only in the case of an RTO. Authors in (Shin et al., 2016) argue that the calculation of RTO for MPTCP flows should include the interface characteristics to improve loss recovery time. A rapid retransmission scheme that uses the monotone increasing packet sequence to detect lost packets was proposed in (Wang et al., 2016). This method was experimentally proven to reduce the packet loss recovery time to less than 2 RTTs for short flows. A less conservative approach for MPTCP based on the TLP mechanism is provided in (Yedugundla et al., 2017) that reduces flow completion time by retransmitting loss probes on alternate subflows in the case of PTO. Results presented in (Yedugundla et al., 2017) are based on synthetic traffic to mimic tail loss at specific position in a flow to evaluate the advantage of TLP over RTO in case of tail losses. The performance of TLP in real world traffic scenarios is not explored.

In this paper, we evaluate the latency performance of TLP in MPTCP with the implementation provided in (Yedugundla et al., 2017). This paper provides a comprehensive evaluation of flow completion times based on real world traffic scenarios using testbed experiments. The experimentation mimics traffic similar to cloud applications when accessed from a multihomed mobile device that supports MPTCP. A short discussion on handling tail losses in MPTCP along with TLP implementation details are provided in the next Section.

4 HANDLING TAIL LOSSES IN MPTCP

The aim of Tail Loss Probe (TLP) (Dukkipati et al., 2013) is to reduce latency of short flows in tail drop scenarios. Tail drops are in general one or more packet losses at the end of a flow. TLP reduces latency by converting retransmission timeouts (RTOs) occurring due to tail losses into fast recovery. If TCP does not receive any ACKs within two RTTs, TLP transmits one packet. The transmitted packet, also called as loss probe, can be a new packet or a retransmission. When there is tail loss, a retransmission timeout is avoided with the ACK from the loss probe triggering FACK/early-retransmit based fast recovery. A single packet loss is also a tail loss with length of tail as one. This is a special case in the TLP specification that adds 200 ms to the PTO accommodating the case of delayed ACKs. Accordingly, the TLP implementation in Linux accounts for the delayed ACK and waits 200 ms before sending a probe. If the computed PTO is larger than RTO due to the added 200 ms, then the minimum of PTO and RTO is considered for the PTO. This can be avoided by tuning the ACKs as discussed in (Rajiullah et al., 2015).

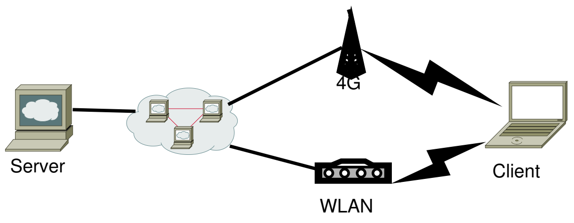

Figure 1: Experimental testbed.

The current MPTCP implementation in Linux uses the TCP TLP per flow. Thus, the probe is sent on the same subflow as the last transmitted packet. This approach, denoted as TLP-STD throughout this paper, is more conservative as it does not use the available other subflows to assist in loss recovery. There is a less conservative approach proposed in (Yedugundla et al., 2017), that sends probe packets on multiple MPTCP subflows. This approach, denoted as TLP-MP throughout this paper, was proven beneficial in certain cases through emulations. In this paper, we investigate this approach further using cloud-based application traffic in a real network scenario. We aim to verify the benefits of using this greedy approach to loss recovery for tail losses. In the next section, we discuss the experiment setup and characteristics of traffic used for the experiments.

5 EXPERIMENT SETUP

The experiment uses traffic from a cloud to mobile end user session for various cloud-based applications. The selected applications are Netflix, Google Maps and Google Docs applications due to their difference in the traffic types. Netflix traffic is mainly flows with large bursts that can fill the congestion window of the first MPTCP subflow allowing the sender to use two paths in MPTCP. Google Docs traffic has very short bursts mostly consisting of one packet. Google Maps traffic has intermediate sized bursts consisting of more than one packet, but not enough to fill the congestion window enough to utilize the second subflow in the MPTCP connection. Traffic is captured using the respective applications for each traffic type.

Table 1: Experiment details.

Traffic	Google Maps, Google Docs, Netflix
Scenario	Path failure to create tail loss
Links	WLAN, 4G in realworld specification
Added delay	10 ms on WLAN

In our experiments, we use traffic data from (Grinnemo and Brunstrom, 2015), which provides a comprehensive information on the cloud application traffic characteristics. Briefly, Netflix traffic has ON-OFF cycles with burst sizes between 500 KB and 1.5 MB. Google Maps and Google Docs traffic have burst sizes of around 100 KB and 400 Bytes, respectively. All the traces used in the experiments are of length 180 secs and 30 traces are considered for each traffic type.

We use a testbed with a client connected to the Internet via two interfaces: 4G and WLAN, and a server connected to the Internet using Ethernet as depicted in Figure 1. The Linux kernel implementation is the most complete MPTCP implementation available, so this experiment setup also allowed us to use the most feature complete version of MPTCP. The server and client run Ubuntu 16.04 with Multipath TCP version 0.89.3. The respective patches are applied and compiled in to the Linux kernel for different implementations of TLP. The WLAN provider and the server are connected to the same Internet service provider. The WLAN link has a round-trip time of 3 ms that is very low to test short flows due to their low burst completion time. Most cloud servers have a little higher delay than 3 ms. In order to make the setup close to a cloud environment, WLAN has an added delay of 10 ms in our experiment setup as mentioned in Table 1. The resulting observed round-trip times are 10-15 ms for WLAN and 70-100 ms for 4G at various times of measurements. WLAN is provided by a router supporting 150 Mbps downlink speed.

The server and client systems run programs that send and receive traffic to mimic the replay of cloud application traffic. As mentioned in (Dukkipati et al., 2013), it is important to localize the burst affected of the loss and measure the flow completion time for that burst. In testbed experiments of this scale, it is difficult to drop a packet deterministically in the same burst. Tail drop on a link is created by creating link failure. We let the server and client establish a MPTCP session and wait until both flows are active. After 80 sec in to the connection, we disconnect the WLAN. This procedure allows us to achieve the same behavior as real world MPTCP handover in WLAN disconnect. Without MPTCP this handover from WLAN to 4G takes nearly 3 sec due to the procedure involving disconnection on one network and reconnecting to the other (Lescuyer and Lucidarme, 2008). This paper evaluates the scenario of path failure where a path fails during the multipath connection to analyze the retransmission behavior using the TLP-STD implementation and the TLP-MP implementation from (Yedugundla et al., 2017). Each experiment is repeated 30 times to achieve confidence intervals with 95%.

6 PERFORMANCE EVALUATION OF TLP VARIANTS

The performance expectations vary with traffic characteristics. To classify traffic broadly, Google Docs and Google Maps traffic can be viewed as low-rate traffic or traffic consisting of short bursts and Netflix traffic as high-rate traffic or traffic consisting of large bursts. Low-rate flows with short bursts do not fill the congestion window, thus utilize a single path until packet loss. On the other hand, high-rate flows have enough data to send to utilize both paths before packet loss.

In our experiments, WLAN disconnection happens at a constant time of 80 secs in to the connection time to create path loss. As the train of bursts in each traffic flow has different sizes, the location at which the loss happens for each traffic is different. In Google Docs traffic, bursts are of one packet length, thus when a loss occurs, it is always in between bursts. This is a scenario of one packet tail drop as discussed in Section 4. For Google Maps traffic the position varies, although significant losses appeared in between bursts as shown in Figure 2. For simplicity, losses in between bursts are counted as to have occurred at the end of a burst. Loss position in Figure 2 illustrates the tail losses occurred with few outstanding packets. In the Netflix traffic case, the bursts are of several packets and loss often occurs after a few packets in to a burst, as shown in Figure 3. The loss position is observed to be in the first 10% of the burst size. With large number of packets within each burst and insufficient time to complete more than two bursts before path failure, the distribution of loss position is skewed towards front of the burst. Loss position is also an indicator of data that is re-injected on the network when an alternate path is used instead of the path used for initial transfer. This re-injection of data is an overhead with MPTCP in general in the case of an RTO.

The burst completion times for Google Docs traffic scenario is depicted in Figure 4. Each bar in the figure depicts the burst completion time for a trace. Google Docs traffic consists of short bursts, often one packet in length, i.e., packet loss should be seen as loss with tail length of one. In this case, TLP-MP

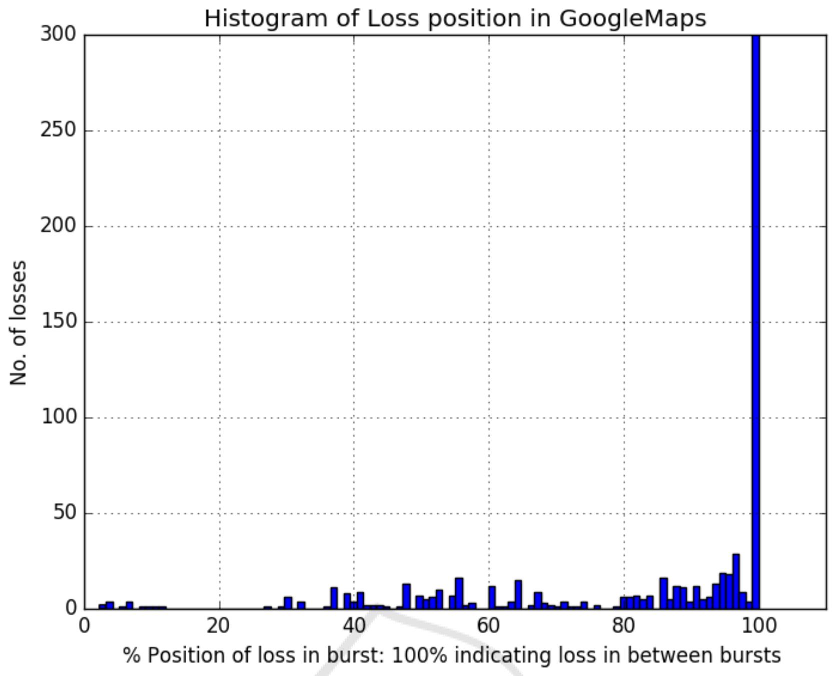

Figure 2: Google Maps loss position in burst.

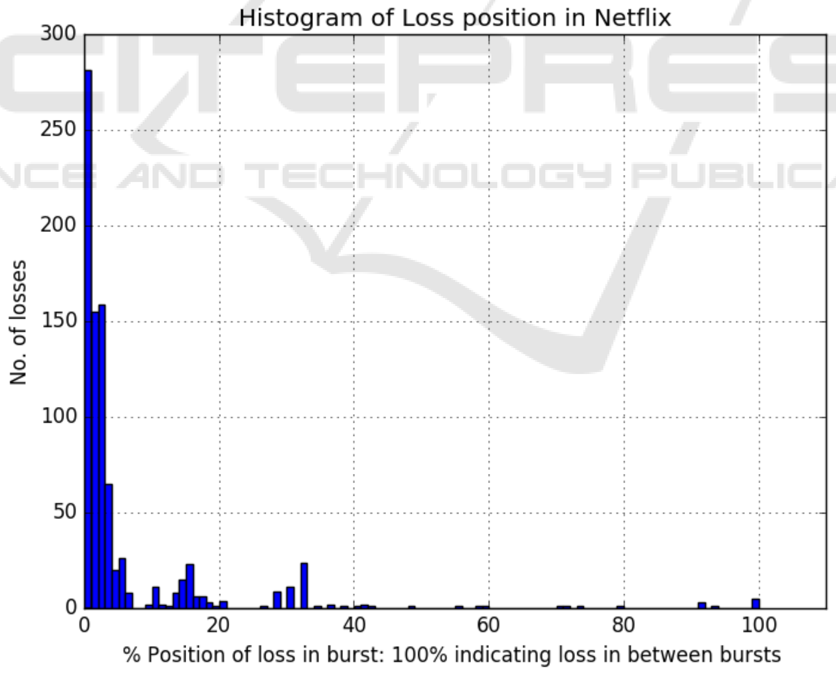

Figure 3: Netflix loss position in burst.

provides a theoretical advantage of 200 ms over retransmission timeout recovery. There is up to 50 percent latency improvement observed for certain traffic traces including the minimum latency gain of 200 ms. Each trace has a different traffic pattern and the gain varies for some traces than others due to the variability in the traffic pattern. However, it is evident from results that TLP-MP outperforms TLP-STD. The loss recovery behavior can be seen in the timing diagram of flows with short bursts in Figure 5. With length

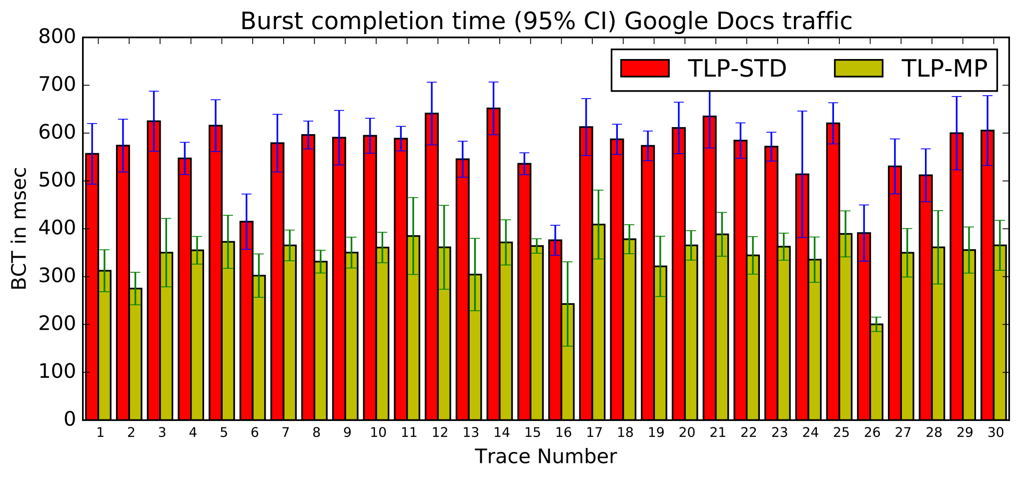

Figure 4: Google Docs traffic.

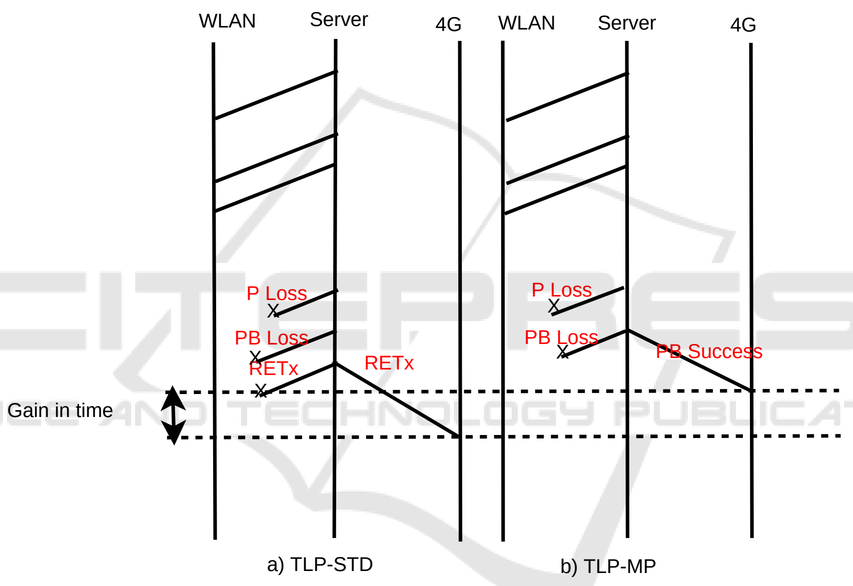

Figure 5: Timing diagram in a flow with short bursts.

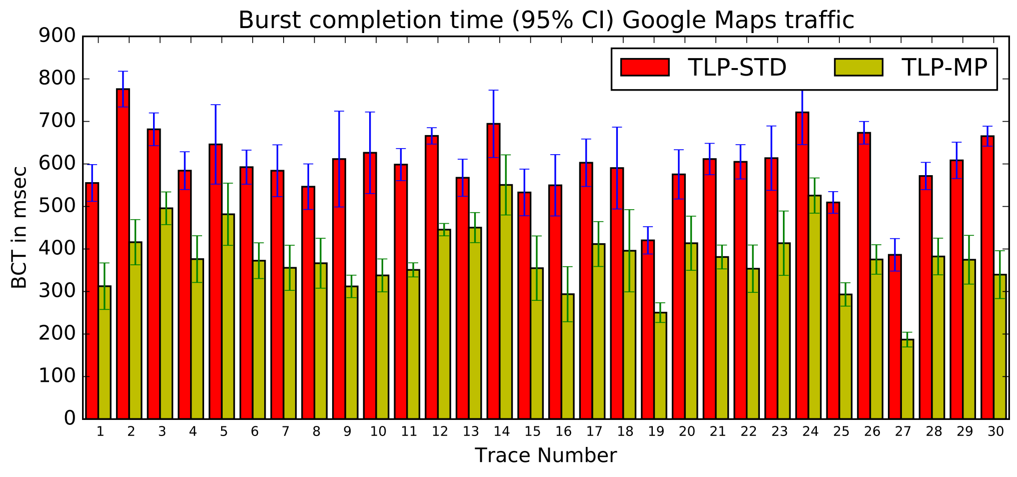

Figure 6: Google Maps traffic.

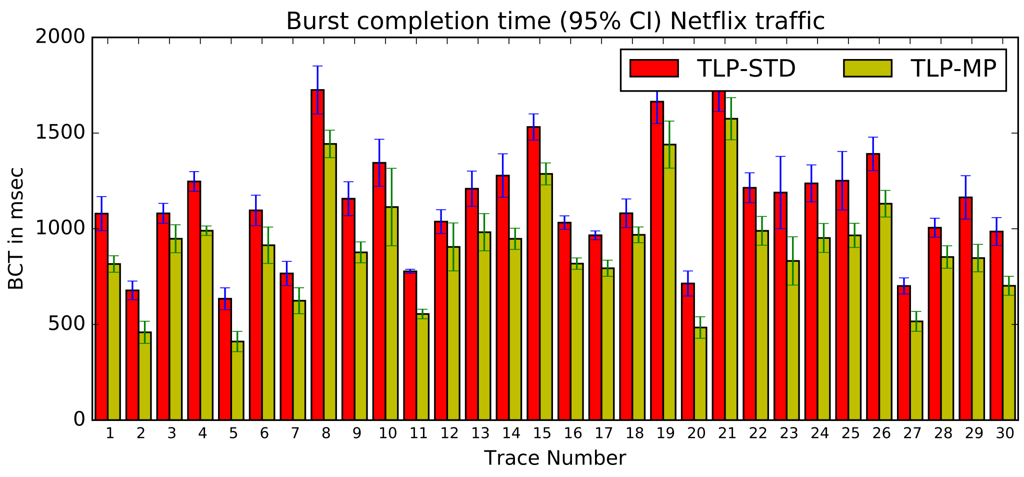

Figure 7: Netflix traffic.

of burst as one packet, in the case of path failure the whole burst is lost. To retransmit the lost burst, the server should wait for an RTO to get loss indication as per original MPTCP implementation (TLP-STD), and retransmit on both subflows incurring more delay as shown in Figure 5a. In the modified MPTCP implementation (TLP-MP), the last transmitted packet is sent as a probe on both subflows and successful retransmission occurs before an RTO as shown in Figure 5b.

Google Maps results in Figure 6 show similar latency performance to that of Google Docs traffic with reduction in flow completion times with TLP-MP. Google Maps traffic consists of bursts of few packets and often loss occurs at the end of the burst or in between bursts in case of path failure as shown in Figure 2. The TLP-STD implementation would retransmit lost packets on both paths after a retransmission timeout. The less conservative TLP-MP implementation would use probe packet on the other path and retransmit other packets faster than original TLP. To ensure TCP flow semantics, MPTCP should always retransmit lost packets on their original path until the time to live (TTL) time of the packet though it is redundant, in this case on a failed path. Network overhead with TLP-MP is same as TLP-STD and minimal with few packets dropped in both scenarios.

Netflix traffic results shown in Figure 7 also indicate improvement over the standard MPTCP implementation. The retransmission behavior of the two implementations for flows with large bursts is shown in Figure 8. The difference between flows with large bursts and flows with short bursts in retransmission behavior is that the former have an active second subflow as there is sufficient data to send. With an RTO, there is a wait for the scheduler to learn about path failure and send the remaining unacknowledged or unsent packets on the other subflow to correct out of order delivery of data at the receiver. In TLP-MP, upon a PTO the last transmitted packet is already sent on the second subflow as probe and the scheduler starts to send new packets on the second subflow.

7 CONCLUSIONS

This paper provides an analysis of tail loss handling with two MPTCP TLP approaches for tail loss and using various cloud application traffic. It provides possible retransmission behavior at packet level on a path failure scenario to understand the perceived gain with a less conservative TLP implementation that triggers a retransmission also on an alternate path in the event of tail loss probe timeout. In the conservative approach, similar behavior is seen only with an RTO. Our testbed experiments using a modified Linux implementation, show that the TLP with less conservative approach in fact improves the burst completion time in all evaluated scenarios and by up to 50 percent. For the path failure scenario, where the TLP retransmission on the primary path never succeeds, this gain comes without any additional overhead. However, in the general loss case TLP-MP may cause additional network traffic from the retransmitted packets. However, this additional traffic is small and subject to normal congestion control.

This study is limited to a path loss scenario that serves the purpose of testing the approach in a specific important scenario. Further, we plan to investigate other loss events that occur in the middle of a packet flow. All the loss recovery optimizations mentioned in this paper use packet counting with DUPACK threshold for loss indication. A recent approach known as Recent Acknowledgment (RACK) (Cheng and Card-

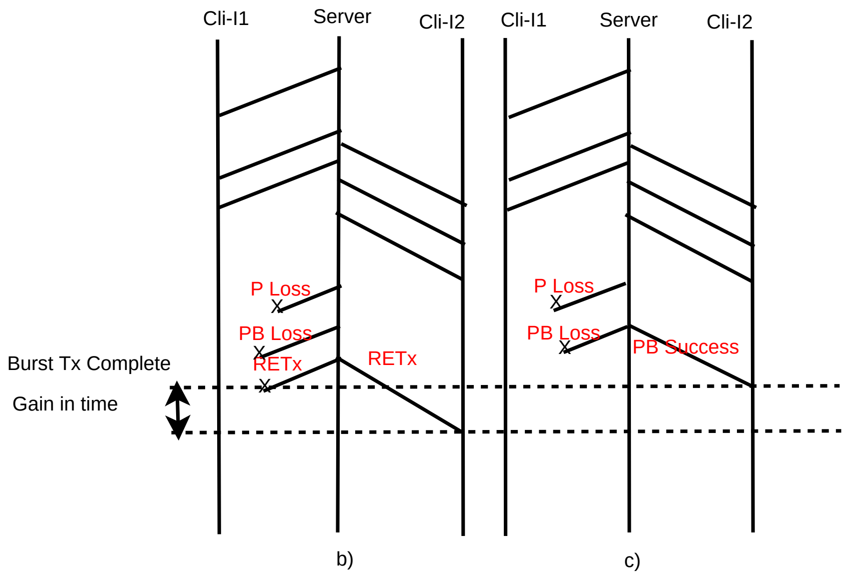

Figure 8: Timing diagram in a flow with large bursts.

well, 2018) uses time-based loss detection algorithm for loss recovery in TCP. We plan to investigate the implications of RACK in MPTCP traffic.

REFERENCES

Allman, M., Avrachenkov, K., Ayesta, U., Blanton, J., and Hurtig, P. (2010). Early Retransmit for TCP and Stream Control Transmission Protocol (SCTP). RFC 5827 (Experimental).

Allman, M., Balakrishnan, H., and Floyd, S. (2001). Enhancing TCP's Loss Recovery Using Limited Transmit. RFC 3042 (Proposed Standard).

Blanton, E., Allman, M., Wang, L., Jarvinen, I., Kojo, M., and Nishida, Y. (2012). A Conservative Loss Recovery Algorithm Based on Selective Acknowledgment (SACK) for TCP. RFC 6675 (Proposed Standard).

Chen, G., Lu, Y., Meng, Y., Li, B., Tan, K., Pei, D., Cheng, P., Luo, L. L., Xiong, Y., Wang, X., and Zhao, Y. (2016). Fast and cautious: Leveraging multi-path diversity for transport loss recovery in data centers. In *2016 USENIX Annual Technical Conference (USENIX ATC 16)*, pages 29–42, Denver, CO. USENIX Association.

Cheng, Y. and Cardwell, N. (2018). RACK: a time-based fast loss detection algorithm for TCP. Internet-Draft draft-cheng-tcpm-rack-04, Internet Engineering Task Force. Work in Progress.

Dong, E., Xu, M., Fu, X., and Cao, Y. (2017). Lamps: A loss aware scheduler for multipath tcp over highly lossy networks. In *2017 IEEE 42nd Conference on Local Computer Networks (LCN)*, pages 1–9.

Dukkipati, N., Cardwell, N., Cheng, Y., and Mathis, M. (2013). Tail Loss Probe (TLP): An Algorithm for Fast Recovery of Tail Losses. draft-dukkipati-tcpm-tcp-loss-probe-01.txt, (Work in Progress).

Flach, T., Dukkipati, N., Terzis, A., Raghavan, B., Cardwell, N., Cheng, Y., Jain, A., Hao, S., Katz-Bassett, E., and Govindan, R. (2013). Reducing web latency: The virtue of gentle aggression. In *Proceedings of the ACM SIGCOMM 2013*, pages 159–170, New York, NY, USA. ACM.

Ford, A., Raiciu, C., Handley, M., and Bonaventure, O. (2013). TCP Extensions for Multipath Operation with Multiple Addresses. RFC 6824 (Experimental).

Frommgen, A., Erbshäußer, T., Buchmann, A., Zimmermann, T., and Wehrle, K. (2016). Remp tcp: Low latency multipath tcp. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–7.

Grinnemo, K. J. and Brunstrom, A. (2015). A first study on using MPTCP to reduce latency for cloud based mobile applications. In *2015 IEEE Symposium on Computers and Communication (ISCC)*, pages 64–69.

Lescuyer, P. and Lucidarme, T. (2008). *Evolved Packet System (EPS): The LTE and SAE Evolution of 3G UMTS*. Wiley Publishing.

Postel, J. (1981). Transmission Control Protocol. RFC 793 (Internet Standard). Updated by RFCs 1122, 3168, 6093, 6528.

Raiciu, C., Paasch, C., Barre, S., Ford, A., Honda, M., Duchene, F., Bonaventure, O., and Handley, M. (2012). How hard can it be? designing and implementing a deployable Multipath TCP. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI'12, pages 29–29, Berkeley, CA, USA. USENIX Association.

Rajiullah, M., Hurtig, P., Brunstrom, A., Petlund, A., and Welzl, M. (2015). An evaluation of tail loss recovery mechanisms for TCP. *SIGCOMM Comput. Commun. Rev.*, 45(1):5–11.

Shin, S., Han, D., Cho, H., Chung, J. M., Hwang, I., and Ok,

D. (2016). TCP and MPTCP retransmission timeout control for networks supporting wlans. *IEEE Communications Letters*, 20(5):994–997.

Wang, W., Zhou, L., and Sun, Y. (2016). Improving multipath tcp for latency sensitive flows in the cloud. In *2016 5th IEEE International Conference on Cloud Networking (Cloudnet)*, pages 45–50.

Yedugundla, K., Ferlin, S., Dreibholz, T., Özgü Alay, Kuhn, N., Hurtig, P., and Brunstrom, A. (2016). Is multi-path transport suitable for latency sensitive traffic? *Computer Networks*, 105:1 – 21.

Yedugundla, K., Hurtig, P., and Brunstrom, A. (2017). Probe or wait: Handling tail losses using multipath TCP. In *2017 IFIP Networking Conference (IFIP Networking) and Workshops*, pages 1–6.