Probe or Wait: Handling tail losses using Multipath TCP

Kiran Yedugundla, Per Hurtig, Anna Brunstrom
Dept. of Computer Science, Karlstad University, Karlstad, Sweden
{name.surname}@kau.se

***Abstract*—Packet losses are known to affect the performance of latency sensitive Internet applications such as media streaming and gaming. Transport protocols recover from packet loss in order to provide reliable end-to-end communication and improve user experience. The efficiency of loss recovery mechanisms influences the completion time of flows, and thus also the application performance as perceived by the end user. In this paper we focus on state-of-the-art loss recovery mechanisms for TCP and Multipath TCP. We use controlled tail loss scenarios to evaluate the performance of loss recovery mechanisms and, based on the observations, we propose an enhanced tail loss recovery mechanism for Multipath TCP, to improve the loss recovery time. Our experiment results, using the Linux Multipath TCP implementation, show consistent end-to-end latency performance improvement in considered scenarios.**

I. Introduction

Multipath TCP (MPTCP) [1] is an experimental standard proposed as an extension to TCP [2]. MPTCP allows devices with multiple interfaces to transmit data simultaneously on all interfaces, thereby improving the end-to-end throughput of connections. Using MPTCP, a connection can have multiple TCP subflows using different interfaces on different routes. Each subflow imparts the functional behavior of a standard TCP flow.

TCP has two mechanisms for detecting and recovering from packet loss: Fast Retransmit (FR) and Retransmission Timeout (RTO) [3]. Fast retransmit is triggered on receipt of a predecided number of duplicate acknowledgements, considering it as an indication of loss. In the case of RTO, a sender waits for the packet loss to trigger a timeout before retransmitting the packet. During such a timeout interval, the sender cannot retransmit any packets. However, with FR the out-of-order packets trigger duplicate acknowledgements. The sender can start retransmission of missing packets without waiting for a timeout. For flows where there are sufficiently many packets to be transferred, FR is quicker than RTO in detecting and recovering packet loss. During long flows, subsequent packets in the flow typically arrive at the receiver after a packet loss and based on the duplicate acknowledgements that they trigger, FR coupled with fast recovery can quickly retransmit the missing data. On the other hand, shorter flows more frequently have to depend on RTO for packet loss detection and recovery, thus incurring more latency. A single packet loss in a short flow may take many RTTs to detect and recover. This scenario is also applicable to the packets at the end of the flow (or tail), of a long flow. Although RTO is reliable even in tail loss recovery, it is not efficient in terms of recovery completion time. Tail losses are one of the major causes of end-to-end latency deterioration in modern networks [3].

ISBN 978-3-901882-94-4

TCP Loss Probe (TLP) [3] is a mechanism to detect and recover from tail losses by employing a loss probe packet sent after a timeout shorter than the RTO. Probe timeout enables short flows to recover packet loss at the tail, by triggering FR. It assumes that other algorithms such as Early Retransmit (ER) [4] and Forward Acknowledgement (FACK) [5] threshold based recovery are enabled. TLP is available in the Linux kernel and enabled by default. Further, evaluations in [6] show that TLP provides significant latency reductions for short flows with one to n-degree tail losses.

MPTCP connections start with one TCP flow to send data over, and may add more flows to the connection if configured to. MPTCP uses a connection level congestion window as well as subflow level congestion windows for each subflow. Data is split across the flows and each data packet is scheduled on one of the subflows. The default MPTCP scheduler in Linux selects the subflow to use based on the shortest RTT. Packet losses in each subflow are detected and recovered in a similar fashion as that of TCP. Loss recovery can be conducted at different levels: the MPTCP connection level (meta level) and at subflow level. If the recovery is handled at meta level, a lost packet may be rescheduled and retransmitted on the available subflow with the lowest RTT. If recovery is handled at the subflow level, the packet is retransmitted on the same subflow. An important part of the loss recovery process at the subflow level is to follow the TCP semantics and retransmit lost packets on the initial path of transfer. However, the MPTCP standard [1] does not specify exactly how the loss recovery should happen in the implementation and which subflow(s) should retransmit the lost packets.

In this paper, we analyze the retransmission behavior of MPTCP in selected tail loss cases to identify the scope for improvement. Based on this analysis, we propose a change in the retransmission strategy of MPTCP to improve the latency performance. We provide related research on MPTCP retransmission mechanisms in Section II, and describe the experimental setup in Section III. We discuss observations from the experiments on state-of-the-art retransmission behavior of TCP and MPTCP in Section IV. Based on the observations, we propose an improvement to the current Linux implementation of TLP for MPTCP in Section V. In Section VI, we provide

an evaluation of the proposed TLP for MPTCP mechanism. Our experimental results indicate a consistent performance improvement of TLP by reducing recovery time in the event of tail losses. Finally, we conclude the paper in Section VII.

II. Background and Related Work

Retransmission schemes are of significant importance in achieving a low flow completion time. Several IETF RFCs such as SACK based recovery [7], Limited Transmit [8] and Early Retransmission [4] have led to improved retransmission strategies for TCP. TLP is part of the various deployable loss recovery mechanisms proposed in [3], suggesting that the evaluated Linux TCP implementation provides room for improvement in several packet loss scenarios.

In a TCP flow, sequence numbers are used to identify lost packets and retransmit them. MPTCP uses two levels of sequence numbers to support efficient data transfer, namely data sequence and TCP sequence. Data sequence numbers are for the end-to-end data transfer and TCP sequence numbers are for individual flow data sequence. There is an association between data sequence numbers and TCP sequence numbers within a subflow. 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.

The target devices of MPTCP contain a degree of asymmetry in their interface characteristics such as 3G or 4G vs WLAN, leading to flows with different delay characteristics. Significant delay differences in available paths can cause receive buffer limitations, as the receiver has to wait for the data sent on a slow path, while buffering the data on the fast path. In such cases, opportunistic retransmission [9] improves the latency by using the fast path to retransmit the data sent on an another path. Authors in [10], try to exploit 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. The MPTCP standard [1] suggests a more conservative approach. MPTCP is designed to exploit the multiple interfaces and often one or more of these could be wireless with completely different characteristics than that of wired interfaces. Authors in [11] argue that the calculation of RTO for MPTCP flows should include the interface characteristics to improve loss recovery time.

This paper analyzes the performance of state-of-the-art loss recovery mechanisms for MPTCP and presents a case for improvement in the tail loss scenario. For understanding the loss recovery and retransmission policies in the MPTCP Linux implementation (version 0.91), we consider several specific cases of tail losses. We use the default settings of the Linux kernel. The Linux implementation of TCP is considered as the baseline TCP retransmission for implementation flexibility and availability of TLP support.

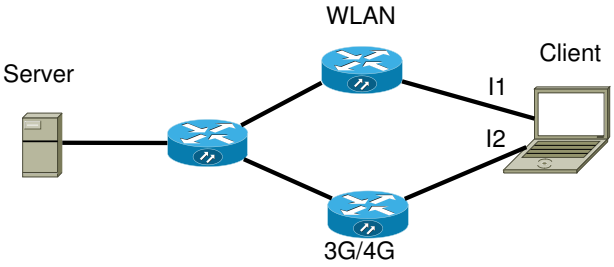


Fig. 1: Topology used for Emulation

Burst Size	80 Packets
Separation Time	2s
Oneway Delay	20 ms-120 ms
Bandwidth	54Mbps
Loss Model	Deterministic

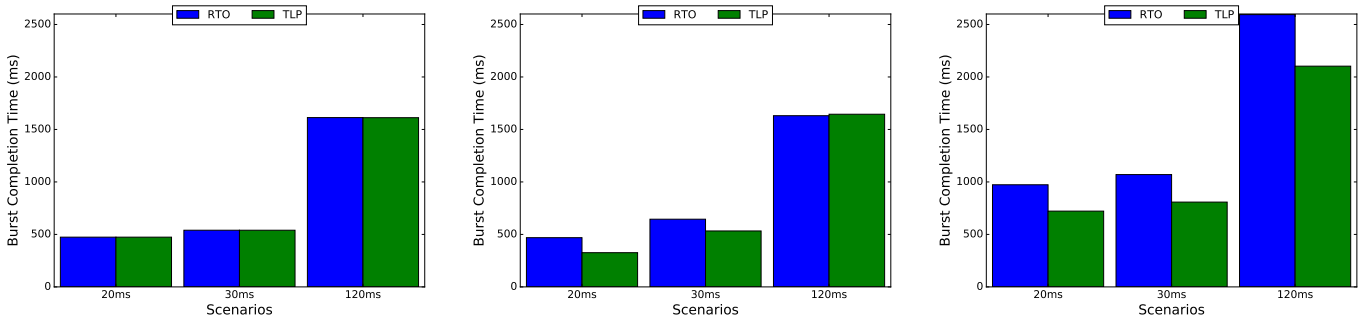
TABLE I: Emulation parameters

III. Experimental Setup

Experiments are run in the CORE emulation platform [12] with a simple topology as depicted in Figure 1. The client node connects to two wireless interfaces 3G or 4G and WLAN and the server node connects to a wired router. CORE uses the Linux networking stack in each node of the topology by using a lightweight virtualization technique known as Linux network namespaces. Characteristics of the connection setup and assumptions about the parameters are provided in Table I. There is no attempt in this study to focus on the effect of link specific characteristics in retransmission performance.

A. Testing retransmission mechanisms with deterministic loss patterns

In order to understand the retransmission behavior of the Linux MPTCP implementation and to reproduce the observed retransmissions, we use a deterministic drop pattern. Losses are generated by associating netem with corresponding interfaces and dropping select packets. This process is simplified by using the KAUNetem tool [13], which allows an experimenter to e.g., drop packets based on their positions in a flow. A MPTCP connection starts with a single TCP subflow and subsequently, one or more subflows are added following an agreement between the client and the server. So one has to wait both in time and packets for the second subflow establishment. We send two bursts of 80 packets each and drop packets at the tail of the second burst at one of the interfaces. This is to ensure that the necessary and sufficient conditions for probe triggering are met and a tail loss probe is generated



(a) Single packet (b) Two packets (c) Single packet and probe loss

Fig. 2: Tail loss scenarios using TCP

on that subflow. We calculate the burst completion time for the second burst that has specified tail losses. The objective of this experiment is to evaluate retransmission mechanisms with tail loss (single packet loss and multiple tail loss) and temporary path loss. Three test cases are considered for the experiments: single packet tail loss and two packet tail loss for tail loss evaluation and single packet along with probe (in TLP) or retransmission (in RTO) loss for path loss evaluation.

In Linux, TCP supports multiple retransmission mechanisms that can be controlled by a system setting. The default setting in Linux enables both ER and TLP, which may not be the case with other operating systems. In this paper, we use TLP to denote the default Linux setting and RTO to denote the use of retransmission timeout only.

The setup has a server and a client with Linux supporting MPTCP running on them. The Client has two interfaces with one way delay on each interface ranging from 20 ms to 120 ms. The intention is to evaluate the effect of symmetry, mild asymmetry and high asymmetry in the link delays. We consider 5 scenarios with delay pairings 20 ms-20 ms, 20 ms-30 ms, 20 ms-120 ms, 30 ms-20 ms, 120 ms-20 ms to understand the effect of delay difference in the performance as shown in Table I.

IV. Observations and Discussion

This section provides the performance analysis of TCP and MPTCP for the considered cases of single packet tail loss, two packet tail loss and retransmission or probe loss.

A. TCP

Retransmission behavior of TCP provides a base case for MPTCP performance as each individual MPTCP subflow is a TCP flow in itself. Our analysis starts with the results using TCP and compares it with that of MPTCP. Figure 2 represents the comparison of TCP burst completion times in the considered scenarios. Path asymmetry is irrelevant in this case as TCP uses a single path.

In the case of single packet tail loss, as shown in Figure 2a, there is no difference in burst completion between RTO and TLP. The retransmission timeout and the probe timeout are the same as there is only one outstanding packet. This is a special case in the TLP specification that adds 200 ms to the PTO accommodating the case of delayed ACKs. The TLP implementation in Linux accounts for the delayed ACK and waits 200 ms before sending a probe. The minimum of PTO and RTO is considered for the PTO. Hence in this special case the value of the retransmission timeout is used in the loss recovery in both RTO and TLP. This can be avoided by tuning the ACKs as discussed in [6].

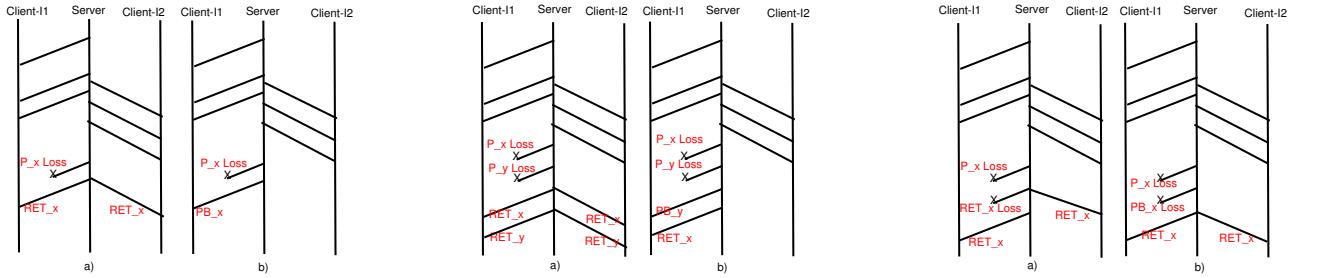
In the case of two packet tail loss, as shown in Figure 2b, TLP performs better than the RTO retransmission for lower delay values. As the delay on the loss path increases, RTO retransmission performs slightly better than TLP. In the 120 ms case, PTO is very close to RTO expiration and in the case of TLP the second packet is recovered using ER which incurs additional delay. In RTO retransmission, the second packet is retransmitted immediately when the acknowledgement for the RTO retransmission is received.

If the probe or retransmission is lost, as shown in Figure 2c, then TLP performs better than RTO retransmission as it waits for one probe timeout and one RTO expiration instead of two RTO expirations, thus avoiding the exponential backoff of the RTO.

B. MPTCP

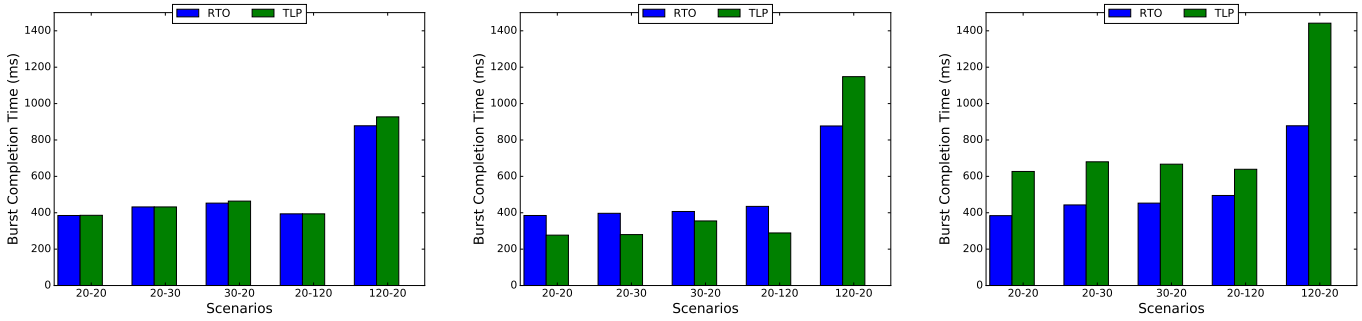
The expected retransmission behavior of MPTCP for RTO retransmission and Linux default settings is shown in Figure 3. The actual pattern might be different in cases where the RTT is larger than 200 ms.

MPTCP uses an RTT based scheduler by default, that uses the lowest RTT path among the available paths to transmit the data until its congestion window is full. Subsequently, the next lowest RTT path is chosen. MPTCP performance is sensitive to asymmetry in paths. The burst completion times on single packet tail loss calculated for each delay configuration and retransmission setting is depicted in Figure 4a. There is no difference in performance between RTO retransmission and TLP for cases when both paths are symmetric or the first path has the lowest RTT. RTO performs better than TLP when the first path is not the lowest RTT path. In such scenarios (30 ms-20 ms,120 ms-20 ms), the RTO retransmission happens in the



(i) Single packet (ii) Two packets (iii) Single packet and probe loss

Fig. 3: Timing diagrams of MPTCP behavior on tail losses with a) RTO b) TLP



(a) Single packet (b) Two packets (c) Single packet and probe loss

Fig. 4: Tail loss scenarios using MPTCP

second path with lowest RTT following the behavior depicted in Figure 3i.

In the case of single packet tail loss, the last outstanding packet is sent as probe and in case of n-degree tail loss, the latest packet is sent as probe to trigger fast retransmit of the packets in between. In order to observe the performance of n-degree tail loss, we consider dropping the last two packets. The results of two packet tail loss are shown in Figure 4b. In this case, TLP performs better than the RTO setting in all scenarios except when packets are lost on a high delay interface. In this scenario, the MPTCP retransmission follows the behavior depicted in Figure 3ii and the delay difference is large enough that the retransmission of the last packet on the alternate path is faster than the probe on the original path.

To further study the performance of TLP in the case when there is a longer path interruption on one path, we tried to drop the probe packet along with the last packet. In this scenario, the RTO setting results in much lower burst completion times. Comparing with the performance of TLP in TCP from Figure 2c, where we saw that the use of TLP improved the burst completion time, TLP in MPTCP did not improve the performance. The reason lies in the way the retransmission and loss probe transmission are carried out in MPTCP. After an RTO, retransmissions occur on both paths. Even if the retransmission fails on one path, the client receives the packet on the other path as shown in Figure 3iii. For TLP, in the current implementation, the loss probe is sent on the actual path of the first transmission but not on both paths. This incomplete impartation of TLP from TCP to MPTCP led to the increase in burst completion time by an RTO for MPTCP.

V. Improvements to TLP for MPTCP

The goal of Tail Loss Probe (TLP) is to reduce tail latency of short flows. It achieves this by converting retransmission timeouts (RTOs) occurring due to tail losses into fast recovery. The Open state of a TCP connection is whenever a sender receives in-sequence ACKs and has not detected any lost segments. If TCP does not receive any ACKs within two RTTs while in the Open state, TLP transmits one packet. The transmitted packet, or loss probe, can be either new or a retransmission. When there is tail loss, the ACK from a loss probe triggers FACK/early-retransmit based fast recovery, thus avoiding a costly retransmission timeout. Our results show that the current TLP implementation for MPTCP helps improving the latency performance in cases when the paths are symmetric or the path where the packet is first transmitted has the lowest RTT. However, it reduces the performance in cases when the first path is not the lowest RTT path or when there is longer path interruptions.

The current implementation of TLP is at the TCP flow level. In the event of RTO timeout, the MPTCP retransmission happens with a re-injection in to the scheduler along with sending the lost packet on the same path. But in the event of Probe timeout, the loss probe packet is being sent only on the same path without being re-injected in to the MPTCP scheduler. We propose a modification to the probe sending

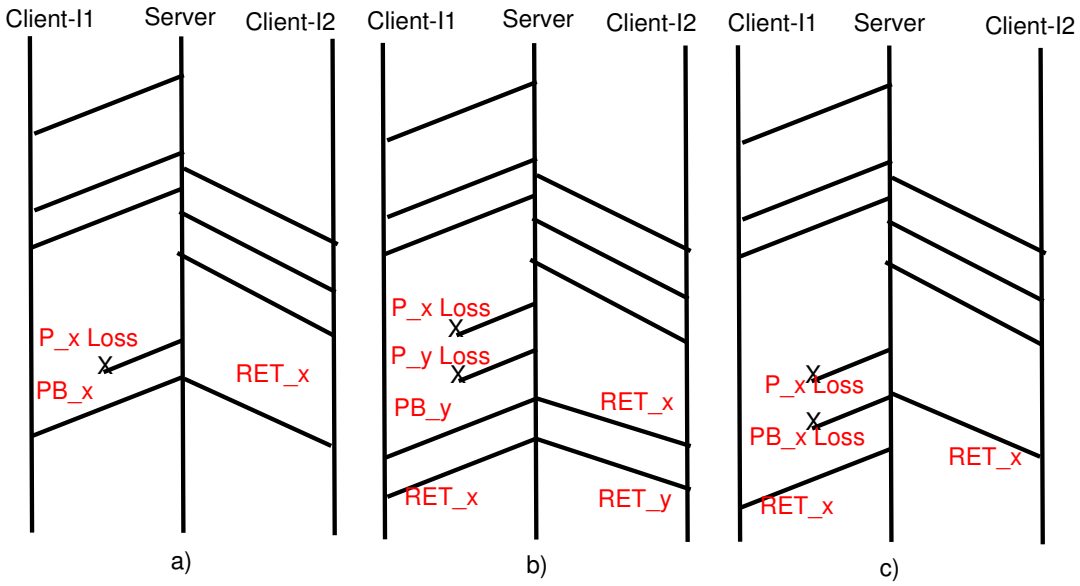
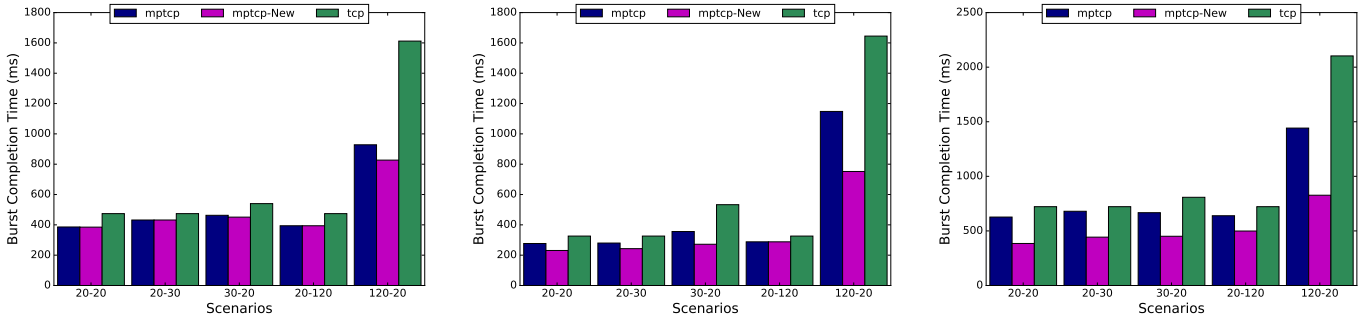


Fig. 5: Timing diagram of MPTCP behavior with proposed improvements with loss of a) Single packet b) Two packets c) Single packet and probe loss



(a) Single packet (b) Two packets (c) Single packet and probe loss

Fig. 6: Performance comparison of MPTCP-New with MPTCP and TCP

mechanism for MPTCP by re-injecting the probe packet in to the MPTCP scheduler in the event of PTO. As the probe timeout is meant for acting faster than the RTO timeout with essentially the same consequence as a retransmission, it is relevant to follow the same process as upon an RTO. It is expected that this modification is useful when the alternate path has lower delay than the path where the packet was originally transmitted or when there is a longer path interruption. The timing diagram depicted in Figure 5 indicates the eventual behavior of modified MPTCP TLP for each of the studied loss scenarios. In the case of single packet tail loss, the lost packet is retransmitted on an alternate path at the same time as the probe is sent on the original path. Since the probe is the last transmitted packet, that is the same as the retransmitted first outstanding packet for single packet tail loss. When two tail packets are lost, the probe sent on the original path is the last transmitted packet and the first outstanding packet is retransmitted on the alternate path. This difference in packet sequence numbers is expected on the two paths due to the nature of the recovery mechanism used on each path. In the case of packet and probe loss on the original path, the alternate path retransmission is the same as the probe packet.

VI. Evaluation of modified MPTCP TLP

Results comparing the burst completion time for new MPTCP TLP with that of TCP and MPTCP TLP for the considered cases are provided in Figure 6. In Figure 6a, MPTCP-New performs better than TCP in all scenarios, performs equal to MPTCP in symmetric scenarios and performs better than MPTCP in asymmetric scenarios with a lower delay on the alternate path. MPTCP-New achieves this improvement by retransmitting the lost packet on the alternate path much before the RTO. It is also observed that there is no performance degradation when the alternate path has higher delay. The probe packet on the original path reaches faster than the retransmitted packet on the alternate path and completes loss

recovery at the MPTCP level. In two packet tail loss scenario as in Figure 6b, MPTCP-New performs better than TCP in all scenarios and better than MPTCP in all scenarios except when the alternate path has a significantly higher delay than the original path, where performance is the same. For this scenario, MPTCP-New achieves this gain as the retransmission on the alternate path recovers the first lost packet and the probe packet on the original path recovers the second lost packet, and thus both lost packets can be recovered in parallel. In the single packet together with probe loss scenario as in Figure 6c, MPTCP-New performs better than both MPTCP and TCP in all scenarios with PTO triggering the retransmission on the alternate path much earlier than the RTO.

VII. Conclusions

This paper provides a short comparative analysis of retransmission behavior with TCP and MPTCP protocols using various loss patterns for tail loss. The three studied cases of tail loss together with deterministic loss patterns allow us to dissect the retransmission behavior at packet level to understand possible improvements. We propose a less conservative approach to trigger retransmission on an alternate path in the event of tail loss probe timeout, which is otherwise triggered only with an RTO. Our emulation experiments using a modified Linux implementation, show that the proposed approach in fact improves the burst completion time in most cases and equals the existing implementation in other cases. The proposed approach improves the performance when the alternate path has lower delay than the original loss path and the advantage increases with degree of asymmetry between the paths. Temporary path failures can cause probe loss along with packet loss. The approach is very effective in case of probe loss which otherwise incur RTO timeout to trigger retransmission on either path. The latency improvement comes at a cost of additional network traffic from the retransmitted packets. However, this additional traffic is small and subject to normal congestion control. This study is limited to emulated synthetic traffic and static drop patterns that serve the purpose of localizing the loss events. A comprehensive study covering various traffic types and loss patterns is considered for future investigation. Further, we plan to investigate less conservative approaches for other retransmission events.

Acknowledgment

This work was in part funded by The Knowledge Foundation (KKS) through the SIDUS READY project. The views expressed are solely those of the author(s).

References

[1] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses," RFC 6824 (Experimental), Internet Engineering Task Force, Jan. 2013.

[2] J. Postel, "Transmission Control Protocol," RFC 793 (Internet Standard), RFC Editor, Fremont, CA, USA, pp. 1–91, Sep. 1981, updated by RFCs 1122, 3168, 6093, 6528.

[3] T. Flach, N. Dukkipati, A. Terzis, B. Raghavan, N. Cardwell, Y. Cheng, A. Jain, S. Hao, E. Katz-Bassett, and R. Govindan, "Reducing web latency: The virtue of gentle aggression," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, ser. SIGCOMM '13. New York, NY, USA: ACM, 2013, pp. 159–170.

[4] M. Allman, K. Avrachenkov, U. Ayesta, J. Blanton, and P. Hurtig, "Early Retransmit for TCP and Stream Control Transmission Protocol (SCTP)," RFC 5827 (Experimental), Internet Engineering Task Force, May 2010.

[5] M. Mathis and J. Mahdavi, "Forward acknowledgement: Refining tcp congestion control," *SIGCOMM Comput. Commun. Rev.*, vol. 26, no. 4, pp. 281–291, Aug. 1996.

[6] M. Rajiullah, P. Hurtig, A. Brunstrom, A. Petlund, and M. Welzl, "An evaluation of tail loss recovery mechanisms for tcp," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 1, pp. 5–11, Jan. 2015.

[7] E. Blanton, M. Allman, L. Wang, I. Jarvinen, M. Kojo, and Y. Nishida, "A Conservative Loss Recovery Algorithm Based on Selective Acknowledgment (SACK) for TCP," RFC 6675 (Proposed Standard), Internet Engineering Task Force, Aug. 2012.

[8] M. Allman, H. Balakrishnan, and S. Floyd, "Enhancing TCP's Loss Recovery Using Limited Transmit," RFC 3042 (Proposed Standard), Internet Engineering Task Force, Jan. 2001.

[9] C. Raiciu, C. Paasch, S. Barre, A. Ford, M. Honda, F. Duchene, O. Bonaventure, and M. Handley, "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*, ser. NSDI'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 29–29.

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

[11] S. Shin, D. Han, H. Cho, J. M. Chung, I. Hwang, and D. Ok, "Tcp and mptcp retransmission timeout control for networks supporting wlans," *IEEE Communications Letters*, vol. 20, no. 5, pp. 994–997, May 2016.

[12] J. Ahrenholz, "Comparison of CORE Network Emulation Platforms," in *Military Communications Conference (MILCOM)*, San Jose, California/U.S.A., Oct. 2010, pp. 166–171.

[13] J. Garcia and P. Hurtig, "KauNetEm: Deterministic network emulation in linux," in *Netdev 1.1 Conference*, 2016.