ReMP TCP: Low Latency Multipath TCP

Alexander Frömmgen[1] Tobias Erbshäußer[1] Torsten Zimmermann[2]
Klaus Wehrle[2] Alejandro Buchmann[1]
[1]DVS, TU Darmstadt, Germany
{froemmge, erbshaeuser, buchmann}
@dvs.tu-darmstadt.de
[2]COMSYS, RWTH Aachen, Germany
{zimmermann, wehrle}
@comsys.rwth-aachen.de

ABSTRACT

More and more Internet-enabled devices, such as server instances or smartphones, have multiple network interfaces. Multipath TCP (MPTCP) has proven to increase bandwidth for these devices, while remaining compatible with the existing network infrastructure and applications. For interactive applications and services, however, low latency and low jitter often are more important than bandwidth.

To meet these challenging latency requirements, we propose ReMP TCP, an MPTCP extension that redundantly sends data over multiple paths. Exchanging bandwidth for latency, this approach guarantees the lowest possible latency in existing best-effort networks facing packet drops and queuing delays. We show the real world applicability of our approach by integrating ReMP TCP into the MPTCP protocol and present first evaluation results.

1. INTRODUCTION AND APPROACH

Interactive applications and services often have tight latency and jitter requirements. These are challenging to meet on best-effort networks like the Internet. For example, coordination protocols such as consensus protocols provide poor performance if they experience high latency and jitter. So do interactive user facing applications such as cloud based assistance systems (e.g., Apple's Siri), audio communication, real time gaming, and interactive web applications. All of those applications have only moderate bandwidth requirements.

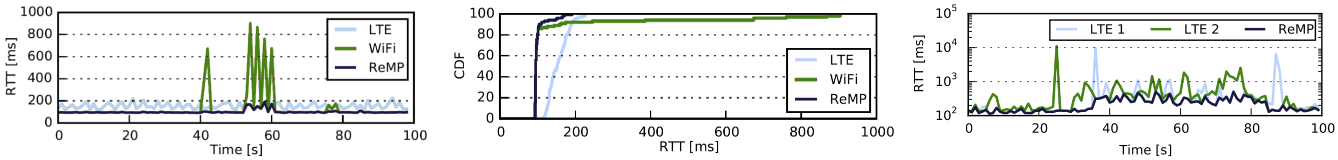
Multipath TCP (MPTCP) [2] increases bandwidth and robustness by using multiple network interfaces and hence multiple paths in parallel [6, 8]. For this purpose, it creates multiple *subflows* to cover the different paths. A major rationale for the design of MPTCP was its compatibility with standard TCP, so that it can be easily deployed in the existing infrastructure and is transparently handled by middleboxes, such as firewalls and NATs. Subflows are dynamically added and removed during operation using TCP options. Furthermore, a variety of fine-tuned congestion controls ensure a TCP-friendly behaviour [4, 7, 11]. As MPTCP provides the same interface and guarantees as traditional TCP, it is well-suited to replace TCP in existing applications.

While MPTCP is optimized for high bandwidth, it barely reduces latency. For applications with tight latency requirements but only moderate bandwidth needs, a modification to MPTCP significantly improves the performance of the application. In this paper, we describe and evaluate an MPTCP flavor that reduces latency and jitter by sending data redundantly over all available paths. This ensures that for every packet the currently fastest path determines the effective end-to-end latency. As queuing delays and packet drops appear unexpectedly, every reaction without redundancy can only act after recognizing the delay. Thus, ReMP TCP gracefully handles unexpected queuing delays, variances in latency, heterogeneous path, and time-varying behavior. It is especially effective in the presence of packet drops, as it avoids expensive retransmissions.

The existing MPTCP protocol provides no special support for latency-sensitive applications. RFC 5897 for MPTCP *Application Interface Considerations* [9] suggests allowing latency requirements to be expressed, but does not indicate how this could be implemented. The idea of reducing latency through redundancy was already applied on other network layers for data center scenarios. Vulimiri et al. [10] use *in-network* packet duplication to cope with changing queuing delays in data centers. RepFlow [12] replicates TCP flows at the *application layer* to reduce flow completion times in data centers. These approaches, however, require an explicit subflow management in the network or ECMP. Transport layer redundancy does not behave TCP friendly and suffers from head-of-line blocking. ReMP TCP

CoNEXT Student Workshop'15, December 01 2015, Heidelberg, Germany

DOI: 10.1145/2842665.2843559



(a) The parallel LTE connection efficiently compensates WiFi packet drops. (b) The cumulated round-trip times for Figure 1a stress the reduced tail latency. (c) The measured RTT with two parallel LTE connections in a train.

Figure 1: Real world measurements of the application layer RTT between Heidelberg and North Virginia.

adopts the benefits of MPTCP, such as dynamic subflow creation, deployability in the real Internet, and a TCP-friendly congestion control.

2. EVALUATION

For the evaluation, we implemented ReMP TCP as scheduler for MPTCP version 0.89.2 [5] for Linux Kernel 3.14.22. We run real world measurements with a notebook and parallel usage of WiFi and/or LTE with the mobile carrier *T-Mobile*. The WiFi access point was connected to a local ISP, for the LTE connection we connected Nexus 5 devices to the notebook. We used a simple TCP echo application to measure the application level RTT between Heidelberg[1] (Germany) and an Amazon ec2 instance in North Virginia. Thus, packet drops and retransmissions have a high impact on the overall latency due to the high propagation delay. Using small messages that fit in one packet, we expect retransmissions to happen mostly due to timeouts. For a detailed analysis, we captured traces on all network interfaces[2] and compute the differences between the packet arrival times on both subflows to determine *what-if* values for pure LTE and WiFi connections.

Experiment 1: Residential WiFi and LTE. We evaluated the RTT with ReMP TCP using WiFi and LTE in parallel in a residential area (Figure 1a). Even though the WiFi connection is almost always faster, the rare packet drops and retransmissions are all compensated by the LTE connection. This is possible, as the RTT of WiFi and LTE show a very low correlation. In the shown measurement, ReMP TCP reduced the average RTT by 27% (Table 1, Exp. 1), the worst case RTT in comparison to WiFi by over 78%, and the worst case RTT in comparison to LTE by over 15%. Regarding the trade-off between minimal average RTT (WiFi) and minimal worst case RTT (LTE), ReMP TCP provides the best of two worlds and ensures minimal average and minimal worst case RTT. The cumulated RTTs (Figure 1b) show the tremendous improvement of the tail latency. Our measurements confirm the results of Chen et al. [1], who reported WiFi packet drop probabilities of 3% (as our figures show RTTs, each point represents two packets), LTE packet drop probabilities of 0.1%, and 15ms latency difference between WiFi and LTE.

Experiment 2: Multiple LTE Devices in the Train. As ReMP TCP is supposed to outperform traditional approaches especially in challenging environments, we repeated the application layer RTT measurements to the server in North Virginia in a train moving with up to 160km/h between Heidelberg and Frankfurt. We connected two Nexus 5 devices to our notebook, thus using two parallel LTE connections. The LTE connections show a lot of packet drops and retransmissions (Figure 1c), which lead to extremely high variances in the end-to-end RTT and a very bad tail latency for single path connections. We were surprised that even though both LTE connections used the same carrier, the packet drops showed nearly no correlation. ReMP TCP more than halves (48.8%) the average latency and reduces the standard deviation by a factor of 19 (Table 1, Exp. 2) for the shown 100 second trace. We repeated the measurements multiple times with single path TCP and ReMP TCP, showing similar behavior.

[1]The CoNEXT conference location 2015.

[2]Traces and source code are available at https://www.dvs.tu-darmstadt.de/research/remptcp.

Table 1: Performance metrics for the end-to-end round trip time of our real world measurements.

		avg [ms]	worst [ms]	σ [ms]
Exp. 1	WiFi	137.28	903.91	152.78
	LTE	154.68	226.43	25.35
	ReMP TCP	100.21	191.13	16.88
Exp. 2	LTE 1	549.90	11061.91	1140.73
	LTE 2	460.10	9721.17	1150.31
	ReMP TCP	224.46	506.21	99.23

3. CONCLUSION AND OUTLOOK

In this paper, we presented ReMP TCP, an MPTCP protocol extension focusing on end-to-end latency. Both real world measurements show the high potential of ReMP TCP, as it more than halved the average round-trip time. For future work, we will analyze the performance of ReMP TCP in more detail, especially regarding flow completion times, and concentrate on efficiency improvements, such as different degrees of redundancy (network coding) and an automatic detection of situations which benefit from ReMP TCP, e.g. the MPTCP mobile handover [3].

Acknowledgments

This work has been funded by the German Research Foundation (DFG) as part of the projects A2/A3 within the Collaborative Research Center (CRC) 1053 – MAKI.

4. REFERENCES

[1] Y.-C. Chen, Y.-s. Lim, R. J. Gibbens et al. A Measurement-based Study of MultiPath TCP Performance over Wireless Networks. In *IMC*, 2013.

[2] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure. TCP Extensions for Multipath Operation with Multiple Addresses. RFC 6824.

[3] A. Frömmgen, S. Sadasivam, S. Müller et al. Poster: Use Your Senses: A Smooth Multipath TCP WiFi/Mobile Handover. MobiCom '15, pages 248–250, New York, NY, USA, 2015. ACM.

[4] R. Khalili, N. Gast, M. Popovic et al. MPTCP is Not Pareto-optimal: Performance Issues and a Possible Solution. CoNEXT. ACM, 2012.

[5] C. Paasch and S. Barre. Multipath TCP in the Linux Kernel available from http://www.multipath-tcp.org.

[6] C. Raiciu, S. Barre, C. Pluntke et al. Improving datacenter performance and robustness with multipath tcp. In *SIGCOMM '11*, pages 266–277, New York, NY, USA, 2011. ACM.

[7] C. Raiciu, M. Handly, and D. Wischik. Coupled congestion control for multipath transport protocols. RFC 6356, Mar. 2011.

[8] C. Raiciu, C. Paasch, S. Barre et al. How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP. In *NSDI*, 2012.

[9] M. Scharf and A. Ford. Multipath TCP (MPTCP) Application Interface Considerations. RFC 6897, Mar. 2013.

[10] A. Vulimiri, P. B. Godfrey, R. Mittal et al. Low Latency via Redundancy. In *CoNEXT*, 2013.

[11] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley. Design, Implementation and Evaluation of Congestion Control for Multipath TCP. NSDI, 2011.

[12] H. Xu and B. Li. RepFlow: Minimizing flow completion times with replicated flows in data centers. In *INFOCOM*. IEEE, 2014.