UNIVERSITÄT OSNABRÜCK

Institut für Informatik
Arbeitsgruppe Verteilte Systeme

Multipath Networking

MPTCP - Scheduling

Henrik Gerdes
MatNr: 424242

Sommersemester
July 1, 2020

Contents

MPTCP - Scheduling

Henrik Gerdes

July 1, 2020

Abstract

An increasing number of mobile nodes provide the ability to utilize multiple network paths. Multi-path connections are able to combine different network paths to enable increased goodput, resilience and seamless handovers between paths. MPTCP promises to do this while being backwards compatible. The performance achieved for latency and goodput depends on the network paths and how packets are scheduled on these paths. The following paper will summarize different scheduling algorithms, show their areas of application and point out their strengths and weaknesses.

1 Introduction

This paper provides an overview of different scheduling algorithms for MPTCP. To follow the content of this paper, knowledge on the ISO-OSI [ISO] reference model and awareness of common scheduling algorithms is advised. The following part ensures a common knowledge base and recaps some fundamentals in the network stack to familiarize the reader with the subject.

1.1 Network communication

The original Transmission Control Protocol (TCP) is a network protocol, specified in RFC 675 (1974) [CDS74], that handles the communication between distributed computer systems. Current extensions of the original revision empower today's internet and every application running on top of it. The most common one being HTTP. With the increase of network devices, especially mobile nodes, the traditional TCP protocol faces some challenges. The connection-orientated approach of TCP was originally designed for wired networks. Although there are approaches like Reno [APS+99] and Vegas [BP95] to improve TCP over radio, these do not use the full potential of modern mobile nodes. While these congestion control algorithms can improve goodput and delay, they are not capable of handling `Connection Aborting REST` or `Data Loss REST` signals. Both signals require a complete new three-way handshake to establish a new TCP connection. Nor are they capable of combining multiple network paths and aggregate bandwidth. These shortcomings require additional waiting time and this impairs the user experience.

1.2 Concepts of MPTCP

Modern mobile nodes provide multiple network interfaces such as WLAN and 3G/4G/5G. Multipath TCP (MPTCP), originally specified in RFC6824 [FRH$^+$13], utilizes multiple network interfaces to increase goodput or offer resilient data connections to improve user experience. Apple was able to decrease network errors of its voice assistant Siri by 80% with MPTCP [BS16]. The usage of MPTCP-Proxys by ISPs allows users to adopt MPTCP without relying on changes by content providers. This enables mobile nodes to aggregate bandwidth, offload data and provide seamless vertical handover.[PCS15]
Unlike the Stream Control Transmission Protocol (SCTP) [SXM$^+$07], an alternative transport protocol with support for independent streams, MPTCP, is usable without requiring configuration changes on network devices within the network layer. These network devices are often referred to as middle boxes. MPTCP uses subflows for its data streams. Each subflow consists of a traditional TCP stream allowing MPTCP usage in every TCP capable network environment.
Although MPTCP has an `MP_PRIO`-option to use a subflow only as a backup, this paper will only cover the simultaneous usage of subflows. Scheduling package on a single subflow is task that traditional TCP is capable of. To gain optimal network utilization, TCPs byte stream calls for in-order package delivery. The usage of multiple TCP streams with possibly different network characteristics demands flexible package scheduling to ensure in-order package arrival. Suboptimal scheduling leads to head-of-line blocking as well as additional interferences described in section 2. These problems may result in a MPTCP user experience similar to or even worse than traditional TCP connection.
The following section will cover the challenges that MPTCP schedulers are bound to encounters and will present various approaches to solve these aforementioned obstacles.

2 MPCTP challenges

The following part lists the most significant challenges that TCP and especially MPTCP connections have to handle. Based on these challenges, metrics for scheduling decisions are introduced. The algorithms in section 3 employ these metrics to make scheduling decisions.

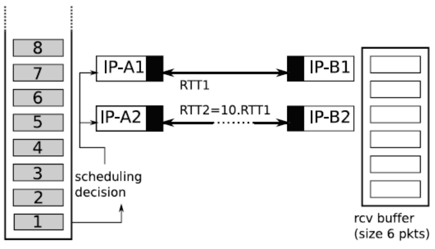
2.1 Asymmetric paths

Every network path has its own network characteristics such as Round Trip Time (RTT), bandwidth and reliability measured in error rates. These characteristics vary based on the network technology used and the location. LAN, WLAN and cellular networks in particular differ in bandwidth, delay and error rates. These errors can consist of disconnects, bit flips, and packets lost [Ela02]. Extensions of TCP have different algorithmic solutions to handle the characteristics introduced by wireless network paths.
By design, MPTCP utilizes multiple network interfaces and each interface uses a different network path with its own characteristics. As a result, MPTCP has to decide how many packets to send via which link and at which moment. This decision influences the package

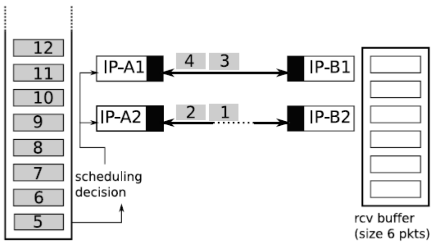
arrival order. A 2016 field study confirms that the use of MPTCP on mobile in fact implies asymmetrical paths [DCBHB16]. The impact of package delivery order is discussed in the next section.

2.2 In order delivery

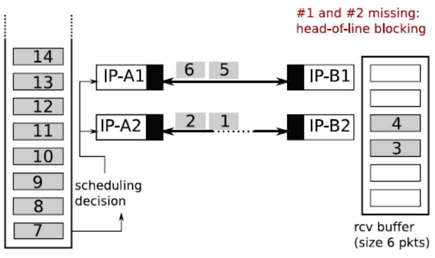
TCP uses incremental sequence numbers to restore the correct data order. MPTCP adds an additional data-sequence number to ensure the correct data order of package that may have been sent over different subflows. The following example shows how asymmetric paths impact transmission time on a MPTCP connection.
Consider two network paths, illustrated in fig. 1, with an $RTT_2 = 10 \times RTT_1$. A Round Robin (RR) scheduler sends data in a circular order over each subflow. Packages send over the faster subflow (RTT_1) arrive 10× earlier in the receiver buffer. Fig. 1c shows head-of-line blocking meaning the buffer cannot be read because previous packages are missing. This also results in receiver-buffer blocking shown in fig. 1d. Because subflow 1 has filled the receiver buffer and thus cannot accept any more packages. The result is that no new packages are sent over the fast subflow.[YFD+16]
Head-of-line blocking introduces additional delay until the receiver buffer can be read. A filled receiver-buffer results in receive-buffer blocking and reduces goodput due to a reduced number of transmitted packages [HGB+18]. These effects may result in transfer time similar to or even worse than a single TCP connection. This demonstrates that RR is not suitable for heterogeneous network paths.



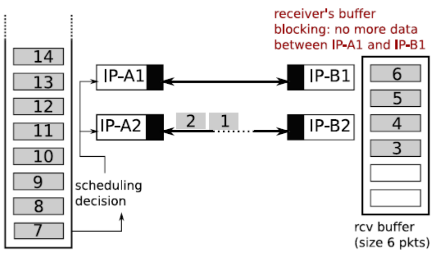
(a) At t_0; asymmetric path $RTT_2 = 10 \times RTT_1$.



(b) At $t_1 = t_0 + \epsilon$; #1 and #2 on path 1; #3 and #4 on path 2.



(c) At $t_2 = t_1 + RTT_1$; #3 and #4 are received but can not be read because #1 and #2 are missing ⇒ head-of-line blocking.



(d) At $t_3 = t_1 + 2 \times RTT_1$; #1 and #2 are not received and the receiver's buffer can not receive #7 ⇒ receiver buffer blocking.

Figure 1: Head-of-line blocking and receive buffer blocking, Source: [YFD+16]

2.3 Use cases and metrics

One aspect to be considered is the area of application for MPTCP. Network applications have different requirements for the network layer. A bulk file transfer benefits from a high throughput. Voice over IP (VoIP) on the other hand relies on a low latency connection. In terms of TCP these are requirements for the Congestion Window (CWND) and RTT. Most MPTCP schedulers use the CWND and RTT as their main metrics for package scheduling. The next section will present different MPTCP schedulers and will explain their algorithms.

3 MPTCP-Schedulers

Although some MPTCP implementations have already been deployed like Apples SIRI or the MPTCP implementation in the Linux kernel, there is no official standard scheduler yet. Furthermore, the Internet Engineering Task Force (IETF) recently (March 2020) approved a new revision of MPTCP specification in RFC 6884 [FRH+20]. The new revision introduces minor changes in the MPTCP options and changes the used hash function. Papers on the impact of these changes are not available yet.
The following section provides an overview of different MPTCP scheduling algorithms. These algorithms are either already in sue or have been proposed for MPTCP.

3.1 Scheduler 1: LRF

The Lowest-RTT-First (LRF) uses, as its name implies, the subflow with the lowest RTT. If the CWND of the subflow is filled, LRF switches to the next subflow with the lowest RTT. This results in a continuous transmission of packages as long any subflow has space remaining in its CWND. This approach aims to maximize throughput. While this is suitable for symmetric paths, its performance in asymmetric paths is limited due to head-of-line blocking. Section 4 provides a detailed performance evaluation.
LRF is the current default scheduler of the MPTCP Linux implementation.[IP 19]

3.2 Scheduler 2: BLEST

The BLock ESTimation (BLEST) scheduler, proposed at the IFIP Networking 2016 Conference, is a MPTCP scheduler specifically designed for asymmetric paths. BLEST introduces a new metric to estimate the amount of blocking that a segment, transmitted over a particular subflow, may cause.[FAMB16]
The scheduler maintains an additional send window on the connection level, above the subflows. BLEST assumes that a segment, sent over a given subflow S, will occupy space in MPTCP's send window ($\mathrm{MPTCP_{SW}}$) for at least $\mathrm{RTT_S}$. The remaining space can be used by a faster subflow F. BLEST estimates the amount of data X that will be sent over F and whether or not it will fit in $\mathrm{MPTCP_{SW}}$. BLEST approximates X by

$$X = MSS_F * \left(CWND + \left(\frac{RTT_S}{RTT_F} - 1\right)/2\right) \times \frac{RTT_S}{RTT_F} \tag{1}$$

If the CWND of F is filled, LRF would switch to the slower subflow S. BLEST is able to skip the slow subflow if $X \times \lambda > |MPTCP_{SW}| - MSS_S * (\mathit{inflight}_S + 1)$. This reduces the risk of blocking. So F can be used multiple times during RTT_S. To address inaccuracies in the calculation of X, a correcting factor λ is used. The correcting factor increases when blocking occurs and decreases in its absence.
On June 22nd, the BLEST scheduler was added to the MPTCP Linux kernel implementation (Version 0.95). [IP 19]

3.3 Scheduler 3: STTF

The Shortest Transfer Time First (STTF) uses a similar approach as LRF. Instead of the RTT, STTF uses the expected transfer time for each subflow as a metric. It also considers the time for enqueued, but not yet transmitted segments. As a result STTF also allows segments to be scheduled to a subflow if its unavailable due to a full CWND. Every time the scheduler is invoked, it reschedules all segments in all subflow send buffers and then releases, handing back to the MPTCP control flow for transmission. In case of a transmission interrupt, such as acknowledgments or changes in the subflows state, the scheduler removes all segments from the send buffers and reschedules them. This allows STTF to react quickly to network changes.[Hay15]
The transmission time is calculated by considering the Smoothed RTT (sRTT), the congestion state and the number of queued segments. The pseudo code for transmission time calculations can be found in the appendix.

3.4 Scheduler 4: ECF

Another approach to towards heterogeneous paths is the Earliest Completion First (ECF) scheduler. ECF uses the subflow F with the lowest RTT if it has space in its CWND. If there is no available space in the subflows CWND, ECF has to decide whether to wait for space in F or to use the second fastest subflow S.
ECF approximates the transmission duration by $RTT_F + \frac{k}{CWND_F} \times RTT_F$ where k is the number of packets to schedule, for the faster subflow. In case $RTT_F + \frac{k}{CWND_F} \times RTT_F \leq RTT_S$, the transmission is completed earlier using subflow F. If RTT_S is smaller than the approximated duration, this is due to the number of scheduled packets k. In this case, ECF decreases transmission time by using F and S to utilizes the bandwidth of both subflows. ECF minimizes the time in which the faster path is not utilized.[LNTG17]

3.5 Additional schedulers

The abovementioned schedulers represent the most common schedulers in the MPTCP stack. They use different metrics and prioritize different parts of package transmission. Nevertheless, it should be mentioned that other kinds of schedulers exist. The Out-of-Order Transmission for In-Order Arrival Scheduler (OTIAS) [YWA14] and Delay-Aware Packet Scheduler (DAPS) [SBL+13] also use the RTT and CWND to schedule packages

on asymmetric paths. The RR scheduler dos not considered neither the RTT nor the CWND. The `redundant scheduler` transmits all packets on all subflows to achieve the lowest possible latency by sacrificing bandwidth [IP 19].
These schedulers do not try to achieve the best possible combination of delay and goodput and will therefore not be considered in the performance evaluation.

4 Evaluation

The evaluation of the schedulers is divided into two parts. The first section details the functional differences and approaches that set these schedulers apart. The second part evaluates their performance, using simulated environments as well as real-world tests.

4.1 Functional comparison

In an asymmetric environment, schedulers like RR and LRF do not achieve optimal performance on neither goodput nor latency. While LRF prioritizes the faster subflow, LRF does not consider the ratio between different subflow characteristics. [HGB+18]
BLEST and STTF both aim for low latency. They are both able to skip a subflow. While BLEST waits for the faster subflow, STTF purely relies on its expected transfer time and the rapid rescheduling for changes in the network paths. The implementation of STTF is considered as complex and the rapid rescheduling as well as the movement of segments require a certain amount of computational resources. Furthermore, regular skipping subflows with high RTT result in lower overall goodput than combined goodput of all subflows.[HGB+18]
ECF is able to improve the utilization of the fastest path. In high packet workloads, ECF is also able to use bandwidth aggregation of several paths. The next section shows how the bandwidth aggregation perform compared to regularly skipping subflows.[LNTG17]

4.2 Simulated and Real-World performance

The following sections evaluate the schedulers performance by ranked latency, the bandwidth provided and real-world performance. The results are drawn from various papers and summarized.

4.2.1 Latency analysis

Fig. 2 shows the results for varying segment sizes over symmetric, asymmetric and highly asymmetric network conditions. The test was performed in a controlled environment (Linux MPTCP V. 0.91.2) and the faster subflow always had an $\mathrm{RTT}_{\mathrm{S}} = 10ms$. It is noticeable that a regular TCP connection is always inferior of a MPTCP at a segment size of 35 and above, except for highly asymmetric paths. This supports the conclusion that LRF is unsuitable for highly asymmetric environments. Schedulers with asymmetric paths in mind,

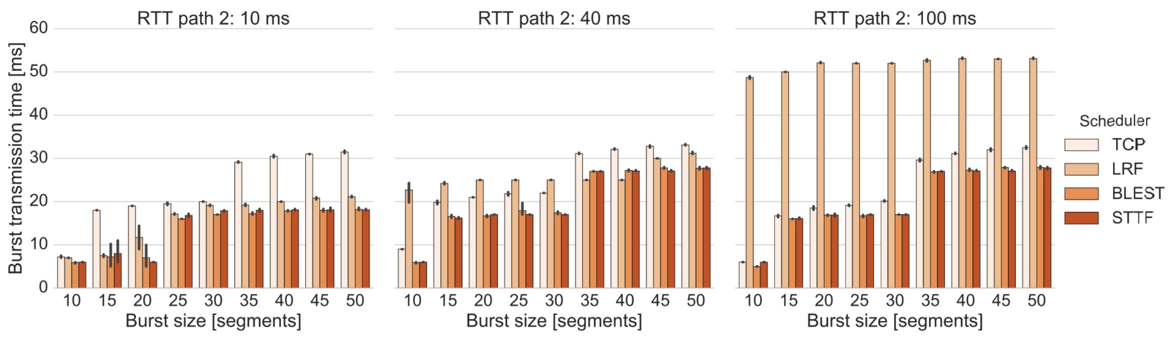


Figure 2: Average transmission time for schedulers, Source: [HGB+18]

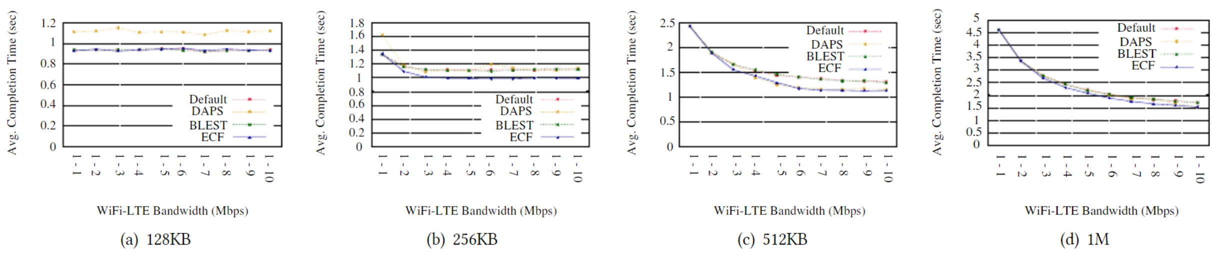


Figure 3: Average Download Completion Time, Source: [LNTG17]

such as BLEST and STTF, outperform LRF as well as regular TCP connections, resulting in lower transmission times overall. This advantage however decreases with greater asynchrony. An analysis on low latency schedulers found ECF to be unsuitable for asymmetric paths.[HGB+18]

4.2.2 Bandwidth analysis

While BLEST and STTF achieve shorter transmission times than LRFin asymmetric paths, their bandwidth advantage over a single TCP connection is minor. This may be due to regularly skipping subflows. Fig. 3 shows the performance of different schedulers for variable bandwidth. Unfortunately, STTF is not included in fig.3.
In order to measure the handling of bandwidth, a WLAN (1 Mbps) path and different LTE paths (1-10 Mbps) were compared. For this test, a bulk file transfer was used. The test was repeated with different file sizes. The average completion time is shown in fig. 3a-d.
The DAPS scheduler, which is not covered in this paper, consistently performs worse than any of the other schedulers, especially for small files. At file sizes above 256KB, BLEST and the default scheduler (LRF) perform similarly. The ECF scheduler always completes the fastest. In asymmetric paths, in terms of bandwidth, ECF completes up to 0.4s earlier at file sizes between 256KB and 512KB. The reason for this could be that ECF utilizes both subflows instead of frequently skipping the slower path.
A test for latency analysis showed that LRF, BLEST and STTF are all able to improve

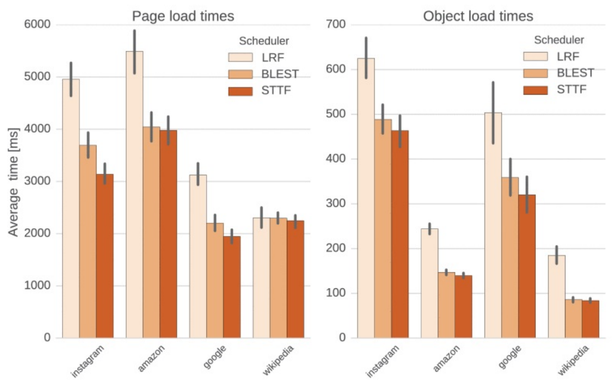


Figure 4: Page load and object times for HTTP2, Source: [HGB+18]

goodput of a MPTCP connection as long as asynchrony was $\frac{\mathrm{RTT}_2}{\mathrm{RTT}_1} \leq 4$. A higher asynchrony resulted in dramatically degraded bandwidth. LRF and BLEST were only able to deliver 10 to 40% of the original TCP bandwidth. While STTFs bandwidth also decreased, it was still able to provide 60% of the original TCP bandwidth. [HGB+18]

4.2.3 Real-World analysis

Test frameworks can emulate various environments, but they are unable to fully replace real workload scenarios. Real-world web browsing measurements are shown in fig. 4a. The mobile node used a WLAN interface with an average $\mathrm{RTT}_{WLAN} \approx 25ms$, as well as a 3G interface with an average $\mathrm{RTT}_{3G} \approx 75ms$. Tests were run with the MONROE mobile broadband (MBB) network measurement platform. Client and server were equipped with a MPTCP capable Linux kernel (V. 0.91.2). The test measured the average page and object load time for popular websites over a period of 30 repetitions.[HGB+18]
The results in fig. 4a show, that both BLEST and STTF outperform LRF. The advantages are particularly noticeable on larger pages like Amazon and Instagram. These measurements are supported in a practical study in which the advantages of MPTCP are shown, especially in regard to larger data amounts and longer connection times [DCBHB16].
It is noticeable that the reduced object load time does not transfer to an improved page load time for Wikipedia. This is due to the dependency structured model of Wikipedia and the burst-like transmission of individual objects.
A practical test with the ECF scheduler showed that it was able to improve the load time for the CNN web page by 26% compared to LRF. In terms of goodput, ECF was able to provide a video streaming bandwidth of 7.79 Mbps while LRF provided 6.72 Mbps, an improvement by 16%. The tests were run with a publicly available WLAN hotspot and a LTE connection by AT&T. [LNTG17]
A direct real-world performance evaluation of all four schedulers within the same test setup is non-existent. The increase of deployed proxies servers and devices, will continue to provide data for future evaluation.

5 Conclusion

The previous sections summarized the current state of MPTCP, explained the challenges of package scheduling and provided an overview of common schedulers tackling these challenges. The evaluation in section 4 showed that the default LRF scheduler is not suitable for MPTCP applications in asymmetric environments. The fact that BLEST was added to the V. 0.95 release of the MPTCP Linux implementation confirms this statement.
The evaluation also showed that the schedulers prioritizes different performance aspects. While BLEST and STTF both improve latency, BLEST suffers from degrading bandwidth performance and STTF is considered computationally intensive. ECF on the other hand is able to improve bandwidth, making it suitable for video streaming but not for latency sensitive applications. These results show that the area of application of MPTCP has a significant impact on which scheduler should be used.
Unlike other multi-path solutions, MPTCP is backwards compatible and requires no configuration changes on middleboxes. Thanks to the MPTCP Linux implementation it is easy to test different schedulers. It also allows seamless integration within Android. By setting up proxy servers by ISPs, companies or universities, larger tests are possible and MPTCP deploys could spread quickly. Data from these future deployments will make it possible to make the best selection of a scheduler for its respective purpose.

Appendix

A Abbreviations

BLEST	BLock ESTimation
CWND	Congestion Window
DAPS	Delay-Aware Packet Scheduler
ECF	Earliest Completion First
HTTP	Hypertext Transfer Protocol
ISP	Internet Service Provider
IETF	Internet Engineering Task Force
LRF	Lowest-RTT-First
MSS	Maximum Segment Size
MPTCP	Multipath TCP
$\mathbf{MPTCP_{SW}}$	MPTCP's send window
OTIAS	Out-of-Order Transmission for In-Order Arrival Scheduler
RR	Round Robin
RTT	Round Trip Time
SCTP	Stream Control Transmission Protocol
sRTT	Smoothed RTT
STTF	Shortest Transfer Time First
TCP	Transmission Control Protocol
VoIP	Voice over IP

B Code for STTF transmission time calculation

```
function TRANSFER_TIME:
    if cwnd_free > 0 and data_to_send < cwnd_free then
        return rtt / 2
    transfer_time = transfer_time + rtt
    cwnd = increase_cwnd(current_cc_state)

    if data_to_send <= max_segments_in_ss then
        transfer_time = transfer_time + rtt * (rounds_in_ss-1) + rtt/2
        return transfer_time
    else if cwnd < ssthresh then
        transfer_time = transfer_time + max_rounds_in_ss * rtt
    if ends_in_ss(data_to_send) then
        return transfer_time

    cwnd = ssthresh
    transmission_time += rtt * (rounds_in_ca - 1) + rtt / 2
    return transfer_time
```

Listing 1: STTF pseudo code, source: [HGB+18]

List of Figures

References

[APS+99] M. Allman, V. Paxson, W. Stevens *et al.*, "TCP congestion control RFC 5681," RFC 5681, 1999, Available: http://rfc-editor.org/rfc/rfc5681.txt.

[BP95] L. S. Brakmo and L. L. Peterson, "TCP Vegas: end to end congestion avoidance on a global Internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, pp. 1465–1480, 1995.

[BS16] O. Bonaventure and S. Seo, "Multipath TCP deployments," *IETF Journal*, vol. 12, no. 2, pp. 24–27, 2016.

[CDS74] V. Cerf, Y. Dalal, and C. Sunshine, "Specification of Internet Transmission Control Program RFC 5681," RFC 675, 1974, Available: http://rfc-editor.org/rfc/rfc675.txt.

[DCBHB16] Q. De Coninck, M. Baerts, B. Hesmans, and O. Bonaventure, "A first analysis of multipath TCP on smartphones," in *International Conference on Passive and Active Network Measurement.* Springer, 2016, pp. 57–69.

[Ela02] H. Elaarag, "Improving TCP performance over mobile networks," *ACM Computing Surveys (CSUR)*, vol. 34, no. 3, pp. 357–374, 2002.

[FAMB16] S. Ferlin, Ö. Alay, O. Mehani, and R. Boreli, "BLEST: Blocking estimation-based MPTCP scheduler for heterogeneous networks," in *2016 IFIP Networking Conference (IFIP Networking) and Workshops.* IEEE, 2016, pp. 431–439.

[FRH+13] A. Ford, C. Raiciu, M. Handley, O. Bonaventure *et al.*, "TCP extensions for multipath operation with multiple addresses," IETF, RFC 8624, 2013, Available: http://rfc-editor.org/rfc/rfc6824.txt.

[FRH+20] A. Ford, C. Raiciu, M. J. Handley, O. Bonaventure, and C. Paasch, "TCP Extensions for Multipath Operation with Multiple Addresses," IETF, RFC 8684, 2020, Available: http://rfc-editor.org/rfc/rfc8684.txt.

[Hay15] D. Hayes, "Report on prototype development and evaluation of end-system, application layer- and API mechanisms," in *RITE scientific deliverables*, no. 317700, 2015, Available: https://riteproject.files.wordpress.com/2015/12/rite_deliverable_1-3.pdf.

[HGB+18] P. Hurtig, K.-J. Grinnemo, A. Brunstrom, S. Ferlin, Ö. Alay, and N. Kuhn, "Low-latency scheduling in MPTCP," *IEEE/ACM Transactions on Networking*, vol. 27, no. 1, pp. 302–315, 2018.

[IP 19] IP Networking Lab. (2019) MultiPath TCP - Linux Kernel implementation. Visited on: 2020-06-05. [Online]. Available: https://multipath-tcp.org/pmwiki.php/Main/HomePage

[ISO] ISO, ISO, "ISO 7498," *Information Processing Systems-Open Systems Interconnection. ISO*, p. 7498.

[LNTG17] Y.-s. Lim, E. M. Nahum, D. Towsley, and R. J. Gibbens, "ECF: An MPTCP path scheduler to manage heterogeneous paths," in *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies.* IEEE, 2017, pp. 147–159.

[PCS15] C. Pollalis, P. Charalampou, and E. Sykas, "HTTP Data Offloading Using Multipath TCP Proxy," in *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, Oct 2015, pp. 777–782.

[SBL+13] G. Sarwar, R. Boreli, E. Lochin, A. Mifdaoui, and G. Smith, "Mitigating receiver's buffer blocking by delay aware packet scheduling in multipath data transfer," in *2013 27th International Conference on Advanced Information Networking and Applications Workshops.* IEEE, 2013, pp. 1119–1124.

[SXM+07] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson, "Stream control transmission protocol (SCTP)," IETF, RFC 4960, 2007, Available: http://rfc-editor.org/rfc/rfc4960.txt.

[YFD+16] K. Yedugundla, S. Ferlin, T. Dreibholz, Ö. Alay, N. Kuhn, P. Hurtig, and A. Brunstrom, "Is multi-path transport suitable for latency sensitive traffic?" *Computer Networks*, vol. 105, pp. 1–21, 2016.

[YWA14] F. Yang, Q. Wang, and P. D. Amer, "Out-of-order transmission for in-order arrival scheduling for multipath TCP," in *2014 28th International Conference on Advanced Information Networking and Applications Workshops.* IEEE, 2014, pp. 749–752.

Declaration of Authorship

I hereby declare that the paper submitted is my own unaided work. I assure that I wrote this paper without using any other means and sources than those specified. As well as the sources used literally or analogously taken from the sources identified as such.

Signature (Henrik Gerdes)

Osnabrück, the July 1, 2020