Demo: Programming Application-defined Multipath TCP Schedulers

Alexander Frömmgen  
KOM – TU Darmstadt  
alexander.froemmgenn@kom.tu-darmstadt.de

Boris Koldehofe  
KOM – TU Darmstadt  
boris.koldehofe@kom.tu-darmstadt.de

Abstract
Multipath TCP is the de facto multipath protocol in today’s Internet. In this demo, we show how the recently proposed ProgMP programming model for Multipath TCP scheduling enables application- and preference-aware Multipath TCP scheduling within the Multipath TCP Linux Kernel. We use ProgMP to systematically derive the specification of a novel scheduler which retains an application-specific acceptable round-trip time and subflow preferences. This scheduler only utilises non-preferred (e.g., metered cellular) subflows if all preferred subflows (e.g., WiFi) do not retain the acceptable upper round-trip time. We further evaluate this novel scheduler by executing the scheduler specification within the ProgMP runtime in the Multipath TCP Linux Kernel.

CCS Concepts • Networks → Transport protocols;

Keywords • Multipath TCP, Scheduling, Specification Language

ACM Reference format:

1 Introduction
Multipath TCP (MPTCP) is the de facto multipath protocol in today’s Internet. MPTCP splits traffic of a single logical connection on multiple TCP subflows, improving throughput and reliability, e.g., by using subflows on different network paths. On the sending side, the MPTCP scheduler maps outgoing packet on subflows. These packets rejoin at the receiver side, where in-order delivery to the application is ensured with global sequence numbers.

Today’s MPTCP implementation in the Linux Kernel [6] provides three schedulers: The default scheduler, denoted minRTT, uses the subflow with the minimum round-trip time (RTT) and not exhausted its congestion window [7, 8]. The round-robin scheduler [7] chooses subflows in a round-robin fashion. Finally, the redundant scheduler sends packets redundantly on all subflows [3, 5]. These schedulers target improved throughput or latency while providing restricted preference-awareness. ProgMP1, a programming model for Multipath TCP scheduling, was recently proposed to enable more flexible, application- and preference-aware Multipath TCP schedulers [4]. ProgMP provides a scheduler specification language, an execution environment in the Multipath TCP Linux Kernel, and an extended socket API for application-defined scheduling.

In this demo paper, we use ProgMP to exemplarily develop a novel application- and preference-aware scheduler. Starting with an illustrating first scheduler, we analyse the dynamic scheduling environment with ProgMP’s PRINT functionality and derive a scheduler which optimizes for an application-defined acceptable upper round-trip time while considering subflow preferences.

2 A First ProgMP Scheduler Example
Figure 1 provides a ProgMP specification for a lowest round-trip time first scheduler (MinRTT), which basically behaves like the MPTCP default scheduler [7, 8]. The specified scheduler determines all subflow candidates which have not exhausted their congestion window by filtering the set of all subflows (sbfCandis in line 1–3). In case at least one subflow is available (line 5), the scheduler PUSHs the first packet from the sending queue S on the subflow with the minimum round-trip time (line 11–12). A detailed presentation of all language primitives and a discussion of the reinsertion queue handling is presented in [2].

3 Analysing Scheduling Behavior
In order to analyse the scheduling behavior and decisions, we extend the previous scheduler with PRINT statements (Figure 2). PRINTs enable a convenient and efficient analysis during the scheduler execution in the Linux Kernel. To reduce the induced overhead, we restrict PRINT operations to occur every 200ms (line 1–3).

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

Middleware Posters and Demos ’17, Las Vegas, NV, USA
© 2017 Copyright held by the owner/author(s). 978-1-4503-5201-7/17/12...$15.00
DOI: 10.1145/3155016.3155017

1https://progmp.net provides a detailed language and framework overview.
The documents distributed by this server have been provided by the contributing authors as a means to ensure timely dissemination of scholarly and technical work on a non-commercial basis. Copyright and all rights therein are maintained by the authors or by other copyright holders, not withstanding that they have offered their works here electronically. It is understood that all persons copying this information will adhere to the terms and constraints invoked by each author’s copyright. These works may not be reposted without the explicit permission of the copyright holder.

4 Preference-aware Scheduling

In many real-world scenarios, users have preferences with regard to the utilized network resources and corresponding subflows. Users of mobile devices, e.g., often prefer WiFi over metered cellular traffic. In datacenters, paths might be associated with different costs. Pure throughput optimizing schedulers might aggregate all subflows regardless of preferences. However, traffic of interactive applications such as voice-based personal assistant systems and most datacenter request-response patterns usually consist of a few packets which fit on a single subflow. For such traffic, customers do not want to waste their metered cellular traffic in case the round-trip time on the preferred subflows is still acceptable. At the same time, significantly higher round-trip times, as experienced with the backup mode in the example after 10 seconds, reduce the user experience. Around 15% of all measurement samples in a huge recent mobile study experienced a significantly higher RTT on WiFi compared with LTE [1], showing the relevance of round-trip time comparisons between WiFi and LTE. As of today, there is no scheduler which schedules packets of interactive applications preference-aware while considering acceptable round-trip times.

We propose a novel application- and preference-aware scheduler which retains an acceptable upper round-trip time. ProgMP enables a convenient specification of such a scheduler (Figure 4). This scheduler only considers backup subflows if no non-backup subflow has a sufficient round-trip time to retain the acceptable upper round-trip time (line 9). The application can inform the scheduler about the acceptable upper round-trip time by setting the R1 register with ProgMP’s extended socket API. Figure 3 (bottom) shows the used subflows for this RTT- and preference-aware scheduler in the Mininet example scenario, given an acceptable RTT of 90ms.

The specified scheduler favors subflow preferences over increased throughput, backup subflows are only used if all non-backup subflows have exhausted their congestion window. ProgMP enables the specification of a scheduler which favors throughput over subflow preferences, i.e., relies on backup subflows in case all other subflows have exhausted their congestion window, by replacing a single expression, SUBFLOWS (line 6) with sbfCandidates.

These examples show that ProgMP enables a convenient specification of application- and preference-aware MPTCP schedulers.

Acknowledgment

This work has been funded by the German Research Foundation (DFG) as part of the projects C2 in the Collaborative Research Center (SFB) 1053 MAKI.

References