# Accelerating Transport Layer Multipath Packet Scheduling for 5G-ATSSS

Fabian Brisch*, Andreas Kassler†‡, Jonathan Vestin†, Marcus Pieska†
Markus Amend§
*Karlstad University, Karlstad, Sweden**, *Deggendorf Institute of Technology, Deggendorf, Germany‡*,
*Deutsche Telekom, Darmstadt, Germany§*
*Email: *fabian.brisch@hs-osnabrueck.de, †andreas.kassler@kau.se, †jonathan.vestin@kau.se, †marcus.pieska@kau.se*
*‡andreas.kassler@th-deg.de §markus.amend@telekom.de*

***Abstract*—Utilizing multiple access networks such as 5G, 4G, and Wi-Fi simultaneously can lead to increased robustness, resiliency, and capacity for mobile users. However, transparently implementing packet distribution over multiple paths within the core of the network faces multiple challenges including scalability to a large number of customers, low latency, and high-capacity packet processing requirements. In this paper, we offload congestion-aware multipath packet scheduling to a smartNIC. However, such hardware acceleration faces multiple challenges due to programming language and platform limitations. We implement different multipath schedulers in P4 with different complexity in order to cope with dynamically changing path capacities. Using testbed measurements, we show that our *CMon* scheduler, which monitors path congestion in the data plane and dynamically adjusts scheduling weights for the different paths based on path state information, can process more than 3.5 Mpps packets at 25 $\mu$s latency.**

***Index Terms*—Multipath Packet Scheduling, P4, MP-DCCP, 5G, ATSSS.**

## 1. Introduction

Mobile users can select a diversity of access technologies for mobile services, including cellular networks (e.g. 2G, 3G, or 4G) or local area wireless networks (e.g. Wi-Fi). In many areas, the coverage and capacity of cellular access need to increase to cope with new services that have more stringent requirements on throughput and latency, such as AR/VR streaming or 4K online gaming. While current 5G cellular networks are being slowly rolled out providing high-speed cellular access to hot spot areas, the global available wireless capacity, resiliency, and robustness of wireless access networks still require significant improvements to match the requirements of such new services.

A significant challenge is to exploit the multiple wireless access networks in a flexible way. Current implementations are limited to utilizing one network interface at the same time, prioritizing connections by the kind of interface (e.g. Wi-Fi over cellular). Utilizing multiple network interfaces at the same time offers various advantages, such as increased reliability and bandwidth of the overall link and a more seamless experience during the failure of one of the wireless links [1]. 5G system architecture proposed the *Access Traffic Steering, Switching, and Splitting* (ATSSS) architecture — *see 3GPP Rel. 16 [2]* that aims to flexibly use and aggregate all locally available wireless access networks. The ATSSS service is provided by an anchor point inside the core network, which enables the User Equipment (UE) to transparently utilize multiple paths over different wireless access networks even if the remote end-host does not support a multipath-capable transport protocol.

However, such an anchor point requires a flexible, high-performance design and implementation in order to enable low latency packet processing functionality for providing ATSSS services at scale, as it needs to concurrently process millions of packets per second for a large number of users. Flexible and high-performance packet processing has recently received attention from emerging programmable data planes [3], which allow a programmer to specify how packets are processed using high-level programming languages such as P4. The forwarding behavior is then compiled to programmable hardware targets that can also include end-host stacks [4]. Indeed, offloading packet processing functionality to re-programmable hardware such as switches (e.g., Tofino), FPGAs (e.g., NetFPGA), or SmartNICs (e.g., Netronome) enables scalability and flexibility while leveraging hardware for low-latency and high throughput packet processing performance. While programmable switches have been proven to be useful for in-network computing [5] or 5G user plane functions [6], accelerating transport layer multipath functionality within programmable data planes is challenging due to the limitations of both the P4 programming language and the programmable hardware. P4 is not a general-purpose programming language and lacks support for e.g. timers, buffering, floating-point operations, and loops which are typically required by multipath transport functionality.

In this paper, we design and implement the main functionality of a 5G ATSSS proxy using the MP-DCCP framework [7] in the P4 language. We compile the pipeline and offload it to a smartNIC, which enables edge deployment of the proxy main packet processing functionality, including packet en- and decapsulation, per UE congestion state tracking of individual access paths, and multipath packet scheduling for downlink traffic. In order to react swiftly to changes in available path capacities, we implement different

*Parts of this work has been funded by the Bavarian State Ministry for Science and Art through the Hightech Agenda (HTA) and by the Knowledge Foundation of Sweden under project grant 20220072-H-01 (DRIVE).*

multi-path packet scheduling algorithms. While simple approaches such as round-robin do not consider the congestion state of individual paths, our more complex schedulers track per-path states, such as round-trip times or transport layer congestion windows, and use this information to infer congestion and dynamically adjust scheduling weights in the data plane itself. Finally, we carry out an extensive evaluation of our design using different traffic scenarios. We show that the most complex *CMon* scheduler can react swiftly to path capacity changes while being able to process more than 3.5 Mpps packets at 25 $\mu$s latency.

## 2. Accelerating Multipath Transport Layer Packet Scheduling in P4 on SmartNIC

### 2.1. Proxy Baseline Design and Implementation

Our 5G ATSSS proxy reimplements the MP-DCCP proxy functionality in P4 by creating multiple tunnels between the UE and the proxy using the MP-DCCP protocol [7], which is a congestion-controlled version of UDP. The proxy can be deployed in the packet core network, e.g. co-located with a User Plane Function (UPF). It combines en- and decapsulation of IP packets into a MP-DCCP connection with a packet scheduler, which distributes the IP packets arriving from the internet into the multipath tunnels over the different access networks towards the multi-homed UE. Figure 1 shows a flowchart of the proxy pipeline. Packets are processed according to their direction (up- or downlink). Uplink packets are identified by the existence of an MP-DCCP header. Information about the congestion state of the individual tunnels/UEs is stored in register arrays. Addresses and sequence numbers are populated from registers during encapsulation. Packets must be identified, giving the correct index for the information about the UE/tunnel. This is done by a lookup on the source IP address of the incoming packet. Because different packet schedulers make decisions on different information, e.g. Round Trip Time (RTT) and packet loss, several register arrays are used to track relevant information, depending on which packet scheduler is used. Such state information is used in the downlink direction to determine which MP-DCCP sub-flow to encapsulate each packet into. Finally, MP-DCCP headers are created for the downlink packets, with all fields except those dependent on the tunnel selected for the packet being populated. The scheduling decision picks one of the individual DCCP tunnels using the packet scheduling algorithms as described in Section 2.2. This also entails writing the correct path sequence numbers and destination address fields into MP-DCCP headers according to the packet scheduling decision.

For the uplink, when receiving an MP-DCCP packet from the UE, an ACK packet is created by cloning the packet to the ingress, truncating it and rewriting headers. The original packet is decapsulated and forwarded to the internet. Also, for the uplink packets, when processing ACKs that come from the UE tunnel endpoint, indicating the reception of a downlink MP-DCCP packet, RTT information and congestion states are updated in the register arrays by the ACK-analysis action per tunnel.

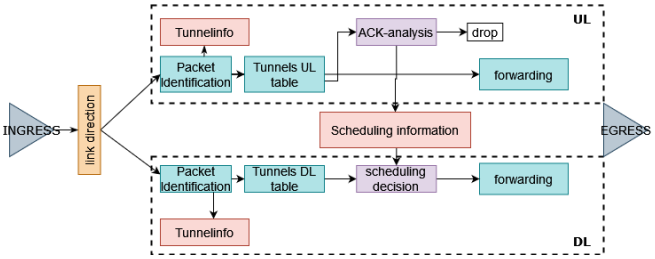

Figure 1: P4 Pipeline Design for Multipath Proxy.

Due to limitations imposed by the P4-framework used, the proxy does not support packet reordering as this would require support for buffering, loops and timers, which is not available in P4. Similarly, floating point and arithmetic operations that are required by more complex packet scheduling algorithms are approximated. Furthermore, access to shared registers was synchronized across packets using mutex locks implemented through Micro-C in the smartNIC in order to avoid race conditions. Our implementation extends [8] by implementing the uplink and several packet schedulers.

### 2.2. P4-based packet schedulers

The challenges for the packet schedulers are to consider the congestion state information of individual MP-DCCP tunnels when deciding over which tunnel endpoint to forward. This is difficult as the available per-path capacity may change rapidly, and UEs may even become temporarily disconnected for some short time (e.g. outage of WiFi connection). Therefore, the scheduler must adaptively shift traffic from one path to another considering per tunnel metrics such as RTT, packet loss, or available bandwidth estimates. As this might be necessary for 100,000s of UEs in parallel, there is a trade-off between scheduler complexity and how well the scheduler reacts to changes in per path state. A fast reaction is desirable in order to avoid unnecessary packet loss or queue buildups if more packets are pushed on a path than the available capacity. We implemented the following scheduling algorithms in P4:

**Weighted Round Robin (WRR):** In order to have a baseline for the evaluation of our more sophisticated schedulers, the WRR scheduler utilizes the P4 built-in random number generation to distribute packets with minimal burstiness over the different tunnels according to the configured weights. While this scheduler does not adapt to capacity changes on the given paths, it is the least complex to implement.

**Dynamic Weighted Round Robin (DWRR):** The DWRR algorithm was implemented in P4 in [9]. It utilizes a modified weighted round-robin algorithm and extends it with congestion detection features. The distribution of packets is defined by two independent weights, allowing a more fine-grained control. The weight adjustment relies on a congestion avoidance approach, increasing the weight of a given path until packets get dropped. This requires an analysis of incoming MP-DCCP ACKs, maintaining a per-path counter for the

maximum sequence number received and information if a packet has been dropped. The algorithm will keep on increasing the weight of a path until it detects packet loss. At that point, it stops increasing the weight of the congested path. A notable deficiency of the described scheduler is the inability of the scheduler to reduce weights when detecting further congestion [9].

**RTTM-based Traffic Distribution (RTTMon):** This scheduler utilizes the reported Round-Trip-Time(RTT) of MP-DCCP packets (and their associated ACKs) to adjust the scheduling weights. On each ACK received from the UE, the proxy calculates the corresponding tunnel RTT and compares it to a target RTT (e.g. 1.5*minRTT). We also implemented the tracking of minRTT in a separate register array. As the scheduler utilizes the same weight implementation as the WRR approach, any weight change corresponds with the decrease in the weight of the other path. Congestion detection is implied by the increase of a given path latency, e.g. due to queues filling up at the bottleneck link. However, as the weight adjustment only occurs on received ACKs, the scheduler cannot detect the outage of a path which results in no incoming ACKs. The scheduler therefore continues to utilize the old weights.

**Congestion based Traffic Distribution (CMon):** This scheduler considers a more complex set of information to infer per path state, including the current path congestion window, minimum RTT, the largest received ACK as well as timing information about when the next adjustment of the congestion window has to happen. If the amount of packets received in a scheduling timeframe does not match a previously set target, the congestion window is reduced. The scheduler allocates packets to the preferred path as long as there is space in the congestion window. Any excess traffic gets allocated to the secondary path.

The P4 implementation of the schedulers is depicted in the control block *scheduling decision* from Figure 1. While WRR and DRR are quite simple to implement, RTMon and CMon require the read and write of several register arrays in the scheduling information block to obtain the necessary information to decide which tunnel to send the current packet over. In addition, they are also more computationally complex.

## 3. Evaluation

For our evaluation, we measure the packet processing complexity of the different schedulers and the achievable processing rate, when accelerating the pipeline on a smartNIC. We use TRex as traffic generator, which replays multiple traffic scenarios against our Device Under Test (DUT). The DUT hosts the one Netronome Agilo CX 2x40G smartNIC, which we connected over a breakout box with 3x10G fiber cables to the TRex machine. We created different traffic profiles by varying packet sizes (64-1024 bytes) and shares between uplink and downlink traffic, representing common scenarios in a 5G-environment [10]. For example, the test scenario 1:0(512B) indicates 100% uplink packets, each one having packets of fixed size 512 bytes, while 0.42:0.52 indicates packets of mixed size representing a typical internet link packet size distribution with 42% uplink and 0.52% downlink traffic shares.

Table 1 shows achievable throughput without packet loss. As can be seen, the packet scheduler complexity has less impact than traffic distribution. This is explained by the fact that if there is significantly more uplink traffic (e.g. 1:0), the pipeline needs to clone many packets to the ingress and truncate them to create ACKs towards the UE. This is a very costly operation, almost reducing throughput by around 35 % compared to when only downlink traffic is served (0:1). Also, the more ACKs are created the more overhead is generated, as ACKs are small in size. On the other hand, it is also clear that the more complex schedulers come with a performance penalty. This penalty increases with the number of downlink packets as the computationally complex scheduling block will be called more often.

TABLE 1: Peak Throughput (mpps)

| Test | WRR | DWRR | RTTMon | CMon |
|---|---|---|---|---|
| 1:0(512B) | 3.0 | 3.0 | 2.9 | 3.0 |
| 1:0(1024B) | 2.9 | 2.9 | 2.8 | 3.0 |
| 0:1(512B) | 5.3 | 5.4 | 4.5 | 4.6 |
| 0.42:0.57 | 4.2 | 4.1 | 3.7 | 3.8 |
| 0.4:0.6 | 4.3 | 4.2 | 3.8 | 3.9 |
| 0.25:0.75 | 4.7 | 4.5 | 4.1 | 4.2 |
| 0.1:0.9 | 4.7 | 4.6 | 4.2 | 4.2 |

Figure 2 shows a violin graph of the packet processing latency as measured by TRex. This includes the sending of the packet from TRex over the fiber cable, the reception in the smartNIC, parsing, ingress and egress processing including packet scheduling and sending it back to TRex. Again, the traffic direction has more impact on packet processing latency than the scheduler complexity. As can be seen, when more uplink packets need to be processed, increased packet cloning and truncating lead to higher processing delay, as queues in the smartNIC tend to get fuller. This negatively impacts the packet processing latency. Comparing the processing latency for a single scenario, it can be seen that the scheduler complexity plays only a major role when there is large number of downlink packets to process.

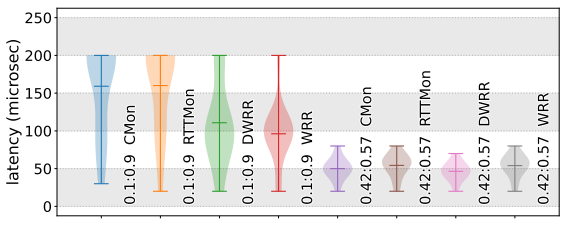


Figure 2: Processing latency for different schedulers.

# References


[1] H. Wu, G. Caso, S. Ferlin, O. Alay, and A. Brunstrom, "Multipath scheduling for 5g networks: Evaluation and outlook," *IEEE Communications Magazine*, vol. 59, pp. 44–50, 4 2021.

[2] 3GPP, "System architecture for the 5G System (5GS)," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 23.501, 07 2021, version 17.0.0.

[3] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[4] N. Foster, N. McKeown, J. Rexford, G. Parulkar, L. Peterson, and O. Sunay, "Using deep programmability to put network owners in control," *SIGCOMM Comput. Commun. Rev.*, vol. 50, no. 4, p. 82–88, oct 2020. [Online]. Available: https://doi.org/10.1145/3431832.3431842

[5] X. Jin, X. Li, H. Zhang, R. Soulé, J. Lee, N. Foster, C. Kim, and I. Stoica, "Netcache: Balancing key-value stores with fast in-network caching," in *Proceedings of the 26th Symposium on Operating Systems Principles*, ser. SOSP '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 121–136. [Online]. Available: https://doi.org/10.1145/3132747.3132764

[6] S. Singh, C. Rothenberg, J. Langlet, A. Kassler, P. Voros, S. Laki, and G. Pongracz, "Hybrid p4 programmable pipelines for 5g gnodeb and user plane functions," pp. 1–18, 08 2022.

[7] M. Amend, E. Bogenfeld, M. Cvjetkovic, V. Rakocevic, M. Pieska, A. Kassler, and A. Brunstrom, "A framework for multiaccess support for unreliable internet traffic using multipath dccp," in *2019 IEEE 44th Conference on Local Computer Networks (LCN)*. Osnabrueck, Germany: IEEE, 2019, pp. 316–323.

[8] R. Alfredsson, A. Kassler, J. Vestin, M. Pieska, and M. Amend, "Accelerating a Transport Layer based 5G Multi-Access Proxy on SmartNIC," in *Würzburg Workshop on Next-Generation Communication Networks (WueWoWas'22)*, 2022, workingpaper, p. 4.

[9] H. Motohashi, P. L. Nguyen, K. Nguyen, and H. Sekiya, "Implementation of p4-based schedulers for multipath communication," *IEEE Access*, vol. 10, pp. 76 537–76 546, 2022.

[10] D. Lee, J. Park, C. Hiremath, J. Mangan, and M. Lynch, "Towards achieving high performance in 5g mobile pakcet core´s user plane function," 2018.