Accelerating a Transport Layer based 5G Multi-Access Proxy on SmartNIC

Rebecka Alfredsson*, Andreas Kassler*, Jonathan Vestin*, Marcus Pieska*
Markus Amend†
*Karlstad University, Karlstad, Sweden**, *Deutsche Telekom, Darmstadt, Germany*†
*Email: *beckaalfredsson@hotmail.com, *andreas.kassler@kau.se, *jonathan.vestin@kau.se, *marcus.pieska@kau.se*
†*markus.amend@telekom.de*

***Abstract*—Utilizing multiple access technologies such as 5G, 4G, and Wi-Fi within a coherent framework is currently standardized by 3GPP within 5G ATSSS. Indeed, distributing packets over multiple networks can lead to increased robustness, resiliency and capacity. A key part of such a framework is the multi-access proxy, which transparently distributes packets over multiple paths. As the proxy needs to serve thousands of customers, scalability and performance are crucial for operator deployments. In this paper, we leverage recent advancements in data plane programming, implement a multi-access proxy based on the MP-DCCP tunneling approach in P4 and hardware accelerate it by deploying the pipeline on a smartNIC. This is challenging due to the complex scheduling and congestion control operations involved. We present our pipeline and data structures design for congestion control and packet scheduling state management. Initial measurements in our testbed show that packet latency is in the range of 25 μs demonstrating the feasibility of our approach.**

***Index Terms*—Multipath, MP-DCCP, 5G-ATSSS, networking, dataplane programming, P4**

1. Introduction

New services result in an increasing demand on the volume of data transmitted over networks and require low and predictable packet latency. However, when using Wi-Fi, connections are frequently interrupted because of the mobility of users, and when the Wi-Fi access point is connected through e.g. ADSL links, bandwidth is scarce. On the other hand, 4G or 5G networks may have more capacity but operator networks risk being congested in dense cities, which increases the desire to offload traffic from cellular networks to Wi-Fi whenever possible. Indeed, mobile handsets come with multiple air interfaces but a coherent architecture to use all of them in parallel in a flexible way is still missing. However, there have been recent efforts to standardize *built-in multipath support* over multiple wireless technologies, including *non-trusted* wireless access like Wi-Fi, in the 5G architecture itself using the first in Rel. 16 specified *Access Traffic Steering, Switching, and Splitting* architecture (ATSSS, see [1]). The *Steering* mode aims to find the best path for a flow according to its requirements, *switching* mode enhance seamlessness from one interface to another without causing interruption in service, while *splitting* mode transparently aggregates all available path capacities. The ATSSS service is provided by an anchor point inside the mobile operator network (see Figure 1) as part of the User Plane Function (UPF), which enables the UE to communicate over multiple access networks, even if the remote server located in the Data Network (DN) does not support a multipath-capable transport protocol. For example, TCP flows may be split at the anchor point (or multipath proxy) and turned into *Multipath TCP* (MPTCP) [2] between the anchor and the user. Non-TCP traffic can be *tunneled over a multipath-capable protocol* [3], e.g. *Multipath DCCP* (MP-DCCP) [4] or *Multipath QUIC* (MP-QUIC) [5] with the Datagram extension [6].

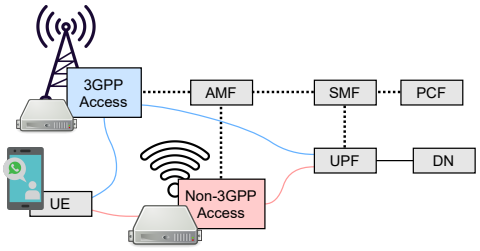

Figure 1: ATSSS architecture within 5G.

The efficiency of the multipath proxy is important for end-users as it impacts the latency and throughput for the UEs. Scalability, energy efficiency and cost is important for operators in terms of numbers of proxies to deploy and their total energy consumption. The data plane of such a proxy must have high performance, process millions of packets for large numbers of users in parallel, at low latency. With the current trend towards network softwarization, implementing such a multipath proxy can be done in either the kernel, user space or can be hardware accelerated. Several measurements show that kernel implementations will have the lowest performance, while user space frameworks (e.g. DPDK [7]) have proven high performance at the expense of high energy demands, as all cores are fully utilized pulling packets from the NIC. Accelerating such a proxy in hardware for optimized performance would require either customized fixed-function ASICs or FPGA implementations that are cumbersome and require a high skillset. With recent evolution

of data plane programmability, we see a dramatic increase in flexibility when hardware accelerating packet processing. Indeed, using programming languages such as P4 allow to specify the packet processing pipeline [8] in high-level languages, while compilers are used to create binary files that are pushed to reprogrammable targets such as switching ASICS, SmartNICs or FPGAs.

In this paper, we leverage data plane programming and design such a multipath proxy based on the MP-DCCP tunneling approach. We implement this design in P4 and evaluate its performance when accelerating it on a smartNIC. This is challenging as the proxy requires packet scheduling over multiple paths to be integrated with transport layer congestion control decisions, which is typically not supported by P4 primitives. The proxy requires significant per UE congestion and scheduling state management. Our implementation leverages external functions and locks for updating congestion control states. We evaluate our prototype in a testbed using packet traces by varying the number of users and packet sizes. Per packet latency is in the order of 25 μs while achieving line rate for downlink processing.

2. Design and Implementation

In this paper, we design and implement the user plane of a 5G ATSSS multi-access proxy based on the MP-DCCP [4] protocol, which extends DCCP [9] with multipath capabilities. The MP-DCCP framework [10], [11] creates multiple DCCP (congestion controlled UDP) tunnels between the UE and the proxy, and transparently distributes traffic from/to the internet over the multiple paths, each one terminated at the UE on a different wireless interface. For packet scheduling and reordering, the MP-DCCP framework offers a pluggable approach. The sending/encapsulation side (the UE for uplink traffic or the proxy for downlink traffic) can leverage different schedulers, including Round Robin (RR), SRTT, Cheapest Path First (CPF). At the receiving/decapsulation side, modular re-ordering modules can be activated including static reordering (e.g. adding constant wait time for outstanding packets) or adaptive reordering (e.g. dynamically adjusting the wait time to cope with path latency diversity). While DCCP has its own (per tunnel) sequence numbers, MP-DCCP adds an additional end-to-end datagram-based sequence number (MP_SEQ option).

The design of MP-DCCP framework faces several challenges when trying to design and implement the proxy functionality in the P4 language. First, the MP-DCCP proxy requires a buffer where incoming packets are stored if the path capacity of all paths are exhausted. Second, the reordering module requires packet buffering and iteration over packet buffers as well as timer support. Third, packet schedulers such as SRTT or CPF require complex per-packet calculations. However, P4 has many restrictions on its expressability and functionality that make it hard to implement all functions that the MP-DCCP framework requires. There is currently no support in P4 for packet buffers, timers, floating point calculations, loops or recursion. Therefore, we resort to a simplified implementation of the proxy, which is based on Round Robin scheduling but does not support packet buffering or reordering. Initial simulations, which are out of scope of this short paper, show that these simplifications still enable effective congestion monitoring, while our future work is centered around designing more advanced packet scheduling based on congestion state of individual paths that will require extensions to our current P4 implementation. In our design, we also assume that the UE has three IP addresses: a generic one that identifies the UE, and one IP address for each physical interface, e.g. Wi-Fi, 5G).

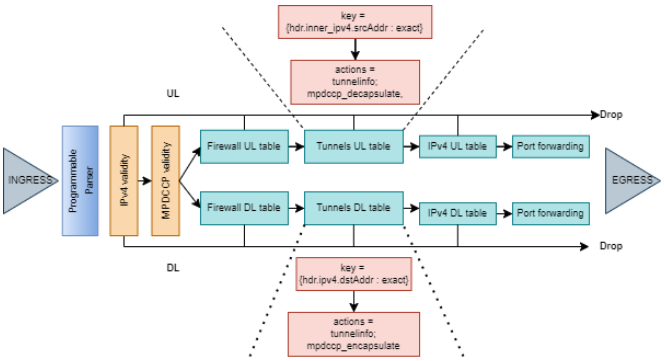

Figure 2: Ingress pipeline processing overview

The first step in the P4 pipeline (see Figure 2) is the parser, which parses required IP, DCCP and multipath extension headers and options using a *lookahead* method. Parsed headers then identify if the packet is a uplink (UL, (MP-DCCP header detected) or downlink (DL, no MP-DCCP header), which trigger different processing paths in the pipeline. For UL packets a firewall UL table is applied followed by a table `tunnels_UL_exact` matching on the IPv4 source address, which identifies the UE. This table has been pre-populated by the control plane to contain static information (e.g. IP and port information required for tunneling) but also index values into register arrays per UE and per tunnel (see Figure 3) that are written to packet metadata and hold stateful information required for packet scheduling and congestion control states; e.g. sequence numbers, last acked packet per path and end-to-end, CWND per path, last used path for scheduler, last sent time per path, last received time, etc. If the packet is a DCCP-ACK packet, we store the DCCP sequence number in a register `ack_reg` using the index received from the table `tunnels_UL_exact`; the packet is then dropped. If the packet is either a DCCP-DataACK or DCCP-Data, the proxy needs to acknowledge the packet. For creating an acknowledgement, the original packet is cloned, the clone is truncated and recirculated so that the egress can transform it into a DCCP-ACK packet that is sent back to the UE after header rewriting. The original packet is decapsulated by removing the DCCP header and the outer IPv4 layer, and routed towards the internet using the `ipv4_UL_exact` table to set the next Ethernet MAC address. The `port_forwarding_exact` table triggers the *port_forwarding* action which sets the egress port.

For downlink (DL) packets, a firewall table is applied first. Then, the proper indices for the registers are identified which store stateful data similarly to the uplink but using `tunnels_DL_exact` and matching on the IPv4 destination address, which identifies the UE in the downlink direction. The packet is encapsulated by adding a DCCP and IPv4 header using information from the metadata. The inner IPv4 header is created by copying the values of the IPv4 header from the original IP packet. The outer IPv4 header is added by setting the IP destination address to that of the cellular or Wi-Fi receiver depending on scheduler state, set the IP source address field to the address of the proxy, and change the protocol field in the IPv4 header to 33 to identify an encapsulated DCCP header. The MP-DCCP source port field is set using *meta.srcport*. The DCCP type field is set to 2 (DCCP-Data), the data offset is set to 7 to represent the DCCP header, padding, and the MP_SEQ multi-path option. The MP_SEQ fields are populated properly. Before any path-specific fields can be populated, the Round Robin (RR) scheduler method is applied as an external function, which decides which path the packet is sent over towards the UE. The scheduler function uses the `scheduling_port` register, if it is equal to 0 path 1 is chosen. The outer IPv4 destination address is then set to *meta.ip_dst_1*, the DCCP sequence number is read from register `inner1_seq_reg` on index *meta.inner1_seq*. The new updated value is now written into that index. The DCCP destination port is set by *meta.dst_port_1* and the value for choosing the next path is written into the `scheduling_port` register. Finally, the packet is routed using the `ipv4_UL_exact` table and the next hop Ethernet MAC destination address is set towards the UE. The `port_forwarding_exact` table is applied and the egress port gets set.

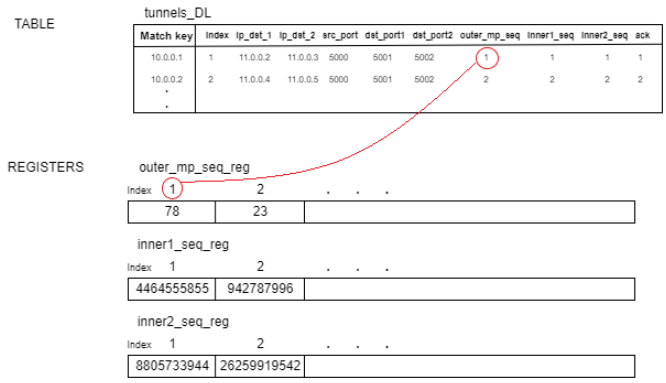

Figure 3: Stateful information layout

In the egress, cloned packets in the uplink are transformed into DCCP-Ack packets that are sent back to UE. The IP source address is set as the proxy, DCCP source port is set using *meta.src_port* and the data offset is set to 9 in order to represent the DCCP header, the acknowledgement number, padding, and the MP_SEQ multi-path option. The DCCP header field type is set to 3 and the acknowledgement number is set to the value of the sequence number. The multi-path sequence number is incremented by one. The packet gets truncated to remove the inner IPv4 layer and payload. Next, a scheduler method decides if the DCCP-ACK packet should be sent using the cellular or Wi-Fi path back to the UE (see above). The outer IP destination address is set using either *meta.ip_dst1* or *meta.ip_dst2*. The DCCP sequence number is either set to the value of `inner1_seq_reg` at index *meta.inner1_seq* or `inner2_seq_reg` at index *meta.inner2_seq*. Lastly the DCCP destination port is set using *meta.dst_port_1* or *meta.dst_port_1* and the updated value for the `scheduling_port` register is set. Finally, stateful registers are updated properly.

3. Preliminary Evaluation Results

We compile the code to Netronome Agilo CX 2x40G SmartNIC (split into 8x10G links) NFP- 4000 SmartNIC platform [12] mounted inside a server with one Intel(R) Xeon(R) Silver CPU with Hyperthreading (HT) enabled for 20 threads at 2.2 GHz each. To evaluate the P4 design with smartNIC offloading, we use another machine of same specification using TREX traffic generator and having an Intel X710 NIC with 2 x 10G SFP+ for sending packets according to static and pre-generated pcap traces to the device under test. Statefulness is pre-populated in the smartNIC to match the generated traces, including tunnel encapsulation metadata and register indices for downlink traffic. We vary the numbers of active UEs and traffic characteristics for downlink flows. Packet sizes range from 128B-1400B, and include mixed traffic of variable sized packets.

In the throughput test, the packet emission rate aims to reach the target aggregate throughput on the downlink link (where MP-DCCP headers are included). As can be seen from Figure 4, achievable throughput increases with packet size and reaches line rate for packets larger than 1024 bytes. For evaluating packet processing latency, we reduce offered rate in second set of experiments where we keep packet size constant to 512B for 10.000 UEs.

Figure 5 shows minimum, maximum and average packet latencies. At low rate, per packet latency (measured time at TREX, which include sending the packets at TREX and receiving it back from DUT) is in the order of 25μs, which increases at higher rate to around 85 μs due to queue buildup. For comparison, we implemented and compared a simple packet in/out pipeline, which resulted in around 5 μs. Additional per packet complexity of the MP-DCCP processing for the downlink is around 20 μs.

4. Conclusion and Future Work

In this paper, we designed, implemented and started a first evaluation campaign for a multiaccess proxy, which is part of the 5G-ATSSS framework. We used P4 and compiled the data plane of the proxy to a smartNIC for hardware accelerating packet processing. The proxy is based on the MP-DCCP framework, which uses multiple tunnels between the proxy and UE to distribute traffic for transparently aggregating the capacity of multiple access networks. Implementation in P4

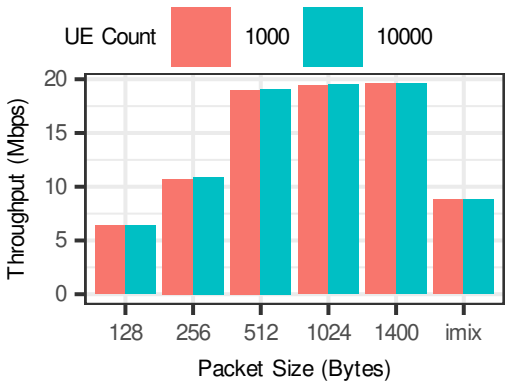

Figure 4: Throughput for different UE and packet size combinations

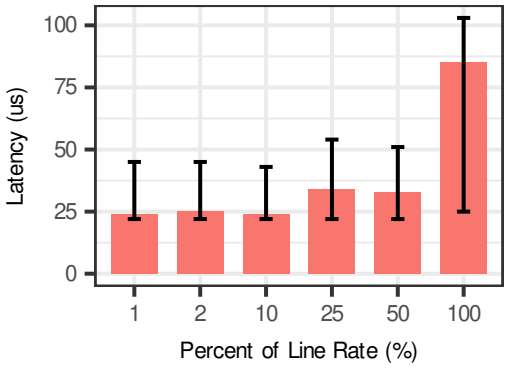

Figure 5: Per packet latency for different offered load

is challenging due to the integrated packet scheduling and congestion control decisions, which we simplified to make a P4 implementation feasible.

Our future work will compare the scheduling performance of this simplified pipeline to that of the kernel implementation of MP-DCCP proxy and demonstrate that the scheduling is effective despite its simplicity. We also aim to evaluate the feasibility of implementing simplified versions of more complex schedulers, with the required congestion control logic, which is still *able to schedule packets effectively* while the design is *still simple enough to be feasible in P4*. Also, we aim to evaluate Uplink and mixed traffic and implementing a full fledged control plane so that dynamic interaction with UEs is possible. Finally, hybrid deployment where parts of the pipeline is implemented on x86 (e.g. buffering and reordering) would be interesting to implement and compare against.

References

[1] 3GPP, "System architecture for the 5G System (5GS)," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 23.501, 03 2021, version 16.0.0.

[2] A. Ford, C. Raiciu, M. Handley, O. Bonaventure, and C. Paasch, "Tcp extensions for multipath operation with multiple addresses," Internet Requests for Comments, RFC Editor, RFC 8684, March 2020.

[3] 3GPP, "Study on Access Traffic Steering, Switch and Splitting support in the 5G system architecture Phase 3," 3rd Generation Partnership Project (3GPP), Technical Report (TR) 23.700-53, 04 2022, version 0.2.0.

[4] M. Amend, A. Brunstrom, A. Kassler, V. Rakocevic, and S. Johnson, "DCCP Extensions for Multipath Operation with Multiple Addresses," Internet Engineering Task Force, Internet-Draft draft-ietf-tsvwg-multipath-dccp-04, 2022, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-tsvwg-multipath-dccp-04

[5] T. Viernickel, A. Froemmgen, A. Rizk, B. Koldehofe, and R. Steinmetz, "Multipath quic: A deployable multipath transport protocol," in *2018 IEEE International Conference on Communications (ICC)*. Kansas City, MO, USA: IEEE, 2018, pp. 1–7.

[6] T. Pauly, E. Kinnear, and D. Schinazi, "An Unreliable Datagram Extension to QUIC," Internet Engineering Task Force, Internet-Draft draft-ietf-quic-datagram-02, Feb. 2021, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-quic-datagram-02

[7] DPDK, "Home - DPDK," https://www.dpdk.org [Online; accessed 7-June-2021]. [Online]. Available: https://www.dpdk.org

[8] 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.

[9] E. Kohler, M. Handley, S. Floyd, and J. Padhye, "Datagram congestion control protocol (dccp)," 2006.

[10] 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)*, Oct 2019, pp. 316–323.

[11] M. Amend, E. Bogenfeld, A. Brunstrom, A. Kassler, and V. Rakocevic, "A multipath framework for UDP traffic over heterogeneous access networks," Internet Engineering Task Force, Internet-Draft draft-amend-tsvwg-multipath-framework-mpdccp-01, Jul. 2019, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-amend-tsvwg-multipath-framework-mpdccp-01

[12] Netronome Systems Inc., "NFP-4000 Theory of Operation," Technical Report, 2016, last accessed: 2022-04-30. [Online]. Available: https://www.netronome.com/static/app/img/products/silicon-solutions/WP_NFP4000_TOO.pdf