On Data Plane Multipath Scheduling for Connected Mobility Applications

Martin Herrmann Amr Rizk
University of Duisburg-Essen
first.last@uni-due.de

***Abstract*—Cooperative, connected, and automated mobility (CCAM) systems depend on reliable communication to provide their service and more crucially to ensure the safety of users. One way to ensure the reliability of a data transmission is to use multiple transmission technologies in combination with redundant flows. In this paper, we describe a system requiring multipath communication in the context of CCAM. To this end, we introduce a data plane-based scheduler that uses replication and integration modules to provide redundant and transparent multipath communication. We provide an analytical model for the full replication module of the system and give an overview of how and where the data-plane scheduler components can be realized.**

I. Introduction and System description

Cooperative, connected, and automated mobility (CCAM) comprises mainly the automated and cooperative road usage of connected vehicles, which is expected to significantly benefit traffic efficiency and safety [1]. The interaction between the connected vehicles is handled by cooperative intelligent transport systems (C-ITS), that provide the necessary services such as hazard notifications and cooperative maneuver coordination to road users. The backbone of a large-scale system such as C-ITS is a stable and reliable communication.

Multiple technologies such as 5G or ITS-G5 are able to provide communication with various grades of reliability. These technologies can be combined and used in parallel to create redundant and thereby more reliable end-to-end communication. Figure 1 provides an overview of the combined usage of multiple transmission technologies in the context of CCAM. The figure shows a scenario where an OBU (ON-board unit) on a vehicle runs the communication functionality for a vehicle-side of a CCAM function, e.g. maneuver coordination. The application on the vehicle is communicating with its counterpart on the Edge (MEC) server. To this end, it is using multiple communication technologies simultaneously, e.g. 5G and mmWave Adhoc WiFi in the 60 GHz band. Evidently, both the OBU and the MEC server require a scheduling function that maps data packets incoming from the application to the respective network interfaces. This mapping can be either forwarding a packet to an appropriate interface or cloning that packet to a subset of available and appropriate interfaces. The counterpart to the scheduler on the sending side, i.e. the uplink direction, is a scheduler entity on the receiver side (MEC) that either forwards the incoming packets or removes packet clones.

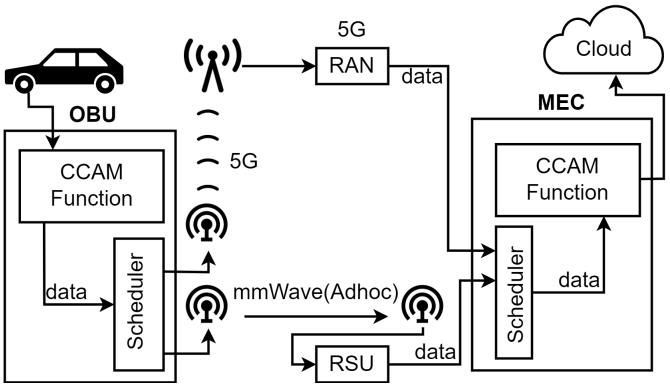

Fig. 1: Reliable end-to-end CCAM communication using a multiple transmission technologies.

As these technologies have different dynamic characteristics the multipath scheduler needs to ensure the right choice of packet-to-technology mapping. This may for example be based on the available communication paths and their status. In addition to the challenge of specifying and optimizing this adaptive multipath scheduler, the question arises of where to implement it. This scheduler could, for example, be implemented in the application layer as Multipath QUIC [2], [3] for flexibility, the kernel space as Multipath TCP [4], [5] for performance. In this work, we opt to show how such a multipath scheduler can be transparently integrated on a lower layer for performance reasons using `tc` [6]. We note that the approach we show here using `tc` can also be ported to XDP [7].

In the following, we first provide a simplified model of the operation of the multipath scheduler as a basis for subsequent optimization. This model is followed by a description of the corresponding data plane implementation.

II. Model

In this section, we describe an analytical model for the multipath scheduling system described in Sect. I for the case of full replication, i.e., every packet arriving at the multipath scheduler is replicated on all paths, and at the receiver the first packet clone is forwarded while the other packet clones are dropped. The following model is a slight variation of the more general replication model from [8]. If the flow is not replicated but rather divided on the different paths an appropriate queueing model is given in [9].

As depicted in Fig. 2, we consider packet arrivals according to a Poisson process with rate λ, where T_i denotes the

This document is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0):
http://creativecommons.org/licenses/by-sa/4.0 This CC license does not apply to third party material (attributed to another source) in this publication.

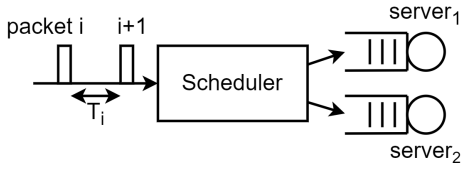

Fig. 2: Queueing model for the (replicating) multipath scheduler: Packets arrive as a Poisson process and are replicated to all available paths that are each modeled as a queue. For each packet the response time is given through the minimum response time over all of its clones.

exponentially distributed inter-arrival time between the ith and $i+1$st packet. The service time of a packet i on path j is given as $X_{i,j}$ which are assumed to be iid exponentially distributed. The scheduler clones every incoming packet to both paths while the receiver forwards every fresh packet to the application and discards its clones upon arrival. The steady-state response time r is known to be distributed as [8]

$$r \overset{D}{=} \max_{n\geq 0}\left\{\sum_{i=1}^{n+1}\min_{j\leq 2}\{X_{i,j}\} - \sum_{i=1}^{n} T_{i-1}\right\} \tag{1}$$

where $\overset{D}{=}$ denotes equal in distribution.

Theorem 1 (special case of [8])**.** *Given a full replication multipath scheduler with Poisson arrivals with interarrival times T_i and iid packet service times on every available path $X_{i,j}$, of a packet i on path j. The tail of the response time distribution given in* (1) *is bounded by*

$$\mathsf{P}\left[r \geq \sigma\right] \leq \mathsf{E}\left[e^{\theta \min_j X_{1,j}}\right] e^{\theta\sigma} \tag{2}$$

with

$$\theta := \sup\left\{\theta \geq 0 | \mathsf{E}\left[e^{\theta \min_j X_{1,j}}\right] \mathsf{E}\left[e^{-\theta T_1}\right] = 1\right\} \tag{3}$$

The theorem above is a straightforward adaptation of a more general statement in [8] where the proof can be found.

Example: Given Poisson arrivals with interarrival times $T_i \sim \exp(\lambda)$ and iid exponentially distributed packet service times on two available paths $X_{i,j}$ with identical parameters $\mu > \lambda$ for stability. Given this replicating multipath scheduler, the packet response time from (1) is, hence, bounded by

$$\mathsf{P}\left[r \geq \sigma\right] \leq \frac{2\mu}{\lambda} e^{-(2\mu-\lambda)\sigma} \tag{4}$$

Interestingly, note that the prefactor above in (4) is larger than one.

We note that the model above is a pure replication model. In principle, it can be extended using more complex characterizations of the traffic (such as martingale envelopes) as well as similar characterizations for the service processes to encompass more realistic traffic and scheduling scenarios.

III. Data plane Multipath Scheduler Realization

This section provides an overview of the multipath scheduling on the data plane for the considered CCAM scenarios. To this end, we use a combination of scheduling programs

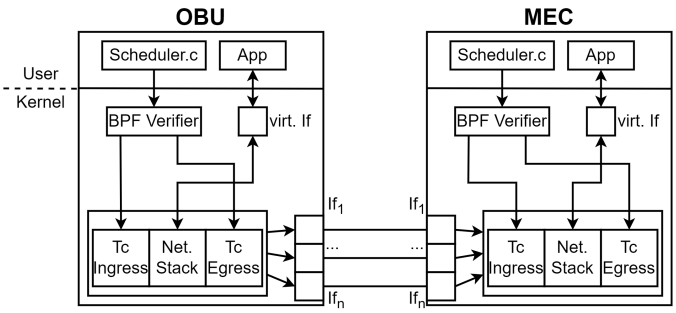

Fig. 3: Simplified model of the considered multipath scheduling system on the data plane of the respective hosts. OBU and MEC stand for on board unit and Multi Access Edge Computing server.

written in C that are loaded as TC egress and TC ingress filters after passing a `bpf` (Berkeley packet filter) verifier. These schedulers map incoming packets to available interfaces according to predefined rules. Additionally, these scheduler programs have access to `bpf` maps, i.e. memory on the respective systems, to save and manipulate a scheduler / packet state.

Figure 3 provides a description of the considered multipath scheduling system on the data plane of the respective hosts. It shows an application that writes data into a virtual interface connected through a scheduler-given TC egress and TC ingress filters to the available interfaces. This scheduler program can be loaded and changed beforehand (or possibly at runtime) as TC filters. The data packet is written or cloned as per the scheduler program on the appropriate interfaces. On the receiver side, the scheduling program is accordingly loaded to execute a complementary task. Given that the sender scheduler program clones data packets (full replication) on the different paths, the receiver scheduler program removes the clones from the data stream. If the sender scheduler is not cloning packets, the receiver scheduler may be set to only forward incoming packets. The receiving scheduler can well be used to obtain monitoring information and also provide packet-level feedback to the sender side.

In the following, we describe the multipath scheduling on the data plane as a pure transparent data packet *replication*. In future work, we will describe the differentiated use of this multipath scheduler beyond replication. The replication system is implemented to utilize two available disjoint paths. The system consists of a sender scheduler that is called replicator, and a receiving counterpart that is denoted integrator.

The first link is used as a shared link between the traffic flows A_1 and A_2. While the second link is used exclusively by traffic flow A_2. The buffer of the shared link uses FIFO scheduling to process the incoming packets. The shared link provides a better connection (i.e. lower average delay, higher average bandwidth) than the second link. To emulate the different characteristics of the two links we use `tc` and `qdisc` [6].

To redundantly send packets of flow A_2 over both link 1

and 2, all packets of flow A_2 must be replicated to both link interfaces. Correspondingly, on the other side duplicate packets of flow A_2 must be handled by dropping them to accommodate applications that are unable to handle these duplicate packets. This replication and dropping or integration can be handled by packet filters.

The replication of flow A_2 to both links is implemented by first identifying and then cloning and redirecting the packets of flow A_2 to the interface of one link and redirecting the packet to the interface of the other link. To this end we use the `bpf` helper function *bpf_clone_redirect*. The replication filter is then compiled and loaded onto the ingress of the scheduler network interface.

The integration of duplicated packets of traffic flow A_2 is done by first identifying the packets belonging to traffic flow A_2 and then comparing the identification information of the packets with the information stored in the `bpf` map. If the information is not already contained in the map, the information is added to the map and the packet is forwarded. Otherwise, the packet is identified as a duplicate and is dropped. Here, we use the `bpf` map type *BPF_MAP_TYPE_LRU_HASH*. The integration filter then is compiled and loaded onto the egress of the scheduler network interface.

IV. Related work on Multipath Communication

Multipath schedulers mapping packets to paths exit in the context of multipath protocols, such as MPTCP [4], [5], [10] and MPQUIC [2], [3]. A logical MPQUIC connection is divided into multiple streams. Each of these streams acts as its own connection with no head-of-line (HOL) blocking between streams. In contrast to MPQUIC, MPTCP multiplexes application data streams on a single connection with possible HOL. The difference between these protocols and our approach is that redundancy is not baked into MPTCP or MPQUIC, and is usually regarded as breaking the transport layer protocol premise. In addition, our approach conducts multipathing on the data plane, while MPQUIC or MPTCP implement multipathing on the user space or in the kernel. This difference allows our scheduler to be faster, while also being more restricted, than a user space implementation would be[1]. Running on the data plane enables our approach to be transparent and allows arbitrary protocols to operate on top.

Multipathing on the data plane exists in data centers where this functionality is set in switches such as the ToR (Top of the Rack) switch. Classically, ECMP multipathing hashes flow headers to consistently map flows to paths. Using the domain-specific language p4 together with p4 programmable switches such as the Tofino switch [11] flows can be steered in granular fashion to use multiple available paths at the same time [12]. The difference to this work is that we set our multipath schedulers into the end-hosts, which is especially suited in the context of 5G hosts as in CCAM scenarios.

[1]The restriction here is actually similar to the restrictions found in some MPTCP frameworks such as in [4]

V. Conclusions

In this paper, we presented a multipath scheduling system description that uses multiple transmission technologies in the context of CCAM. Communication reliability is achieved using a combination of replication and integration of packets which allows the system to provide *transparent* redundant communication between hosts. We reviewed an analytical model for a full replication multipath scheduler and provided a close-form expression given iid exponentially distributed inter-packet arrival times and service times. We describe how and where such a multipath scheduler can be realized on the data plane, using `tc` and `bpf` code. Finally, we briefly discuss other already established methods of multipath communication and how our approach compares to them.

This paper primarily described a preliminary approach to creating a programmable and adaptive multipath transmission system to provide reliable communication for connected mobility applications. Based on this work, we intend to extend and compare our approach and the analytical model to measurements and to other multipath communication methods such as MPQUIC.

References

[1] W. H. Schulz, H. Wieker, and B. Arnegger, "Cooperative, connected and automated mobility: Overcoming the loss of strategic competences by new co-operation models for automotive and telecommunication industries," *Future Telco: Successful Positioning of Network Operators in the Digital Age*, pp. 219–229, 2019.

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

[3] Q. De Coninck and O. Bonaventure, "Multipath quic: Design and evaluation," in *Proceedings of the 13th international conference on emerging networking experiments and technologies*, 2017, pp. 160–166.

[4] A. Frömmgen, A. Rizk, T. Erbshäußer, M. Weller, B. Koldehofe, A. Buchmann, and R. Steinmetz, "A programming model for application-defined multipath TCP scheduling," in *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference*, 2017, pp. 134–146.

[5] S. Barré, C. Paasch, and O. Bonaventure, "Multipath tcp: from theory to practice," in *10th International IFIP TC 6 Networking Conference*, 2011, pp. 444–457.

[6] *tc - Linux Administration and Privileged Commands Manual*.

[7] M. A. Vieira, M. S. Castanho, R. D. Pacífico, E. R. Santos, E. P. C. Júnior, and L. F. Vieira, "Fast packet processing with ebpf and xdp: Concepts, code, challenges, and applications," *ACM Computing Surveys (CSUR)*, vol. 53, no. 1, pp. 1–36, 2020.

[8] F. Ciucu, F. Poloczek, L. Y. Chen, and M. Chan, "Practical analysis of replication-based systems," in *IEEE Conference on Computer Communications, INFOCOM*, 2021, pp. 1–10.

[9] A. Rizk, F. Poloczek, and F. Ciucu, "Stochastic bounds in fork-join queueing systems under full and partial mapping," *Queueing Syst. Theory Appl.*, vol. 83, no. 3-4, pp. 261–291, 2016.

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

[11] A. Agrawal and C. Kim, "Intel Tofino2 - a 12.9 Tbps p4-programmable ethernet switch," in *IEEE Hot Chips 32 Symposium (HCS)*, 2020, pp. 1–32.

[12] C. H. Benet, A. J. Kassler, T. Benson, and G. Pongracz, "MP-HULA: Multipath Transport Aware Load Balancing Using Programmable Data Planes," in *Proceedings of the 2018 Morning Workshop on In-Network Computing - NetCompute*, 2018, p. 7–13.