©IEEE, (2024). *RDA: Residence Delay Aggregation for Time-Sensitive Networking* This is the author's version of the work. To appear in the Proceedings of IEEE/IFIP Network Operations and Management Symposium (NOMS) 2024.

RDA: Residence Delay Aggregation for Time-Sensitive Networking

Chengbo Zhou [*§], Christoph Gärtner [*§], Amr Rizk [¶],
Boris Koldehofe [‡], Björn Scheuermann [*], Ralf Kundel [*]
[*]Technical University of Darmstadt, Germany, {firstname.lastname}@tu-darmstadt.de
[¶]University of Duisburg-Essen, Germany, amr.rizk@uni-due.de
[‡]Technical University of Ilmenau, Germany, boris.koldehofe@tu-ilmenau.de

***Abstract*—Time-Sensitive Networking (TSN) enables deterministic and low-latency communication for real-time applications over Ethernet. That is accomplished by leveraging scheduling and shaping techniques configured for each egress port within the network switches. Although Time Aware Shaper (TAS) is a promising solution for TSN, its adoption often involves substantial complexity. In this work, we propose Residence Delay Aggregation (RDA), a novel asynchronous TSN mechanism that offers dynamic traffic scheduling adapted to the traffic load. Specifically, the proposed RDA mechanism provides upper bound delays similar to other asynchronous TSN mechanisms while improving the flexibility of traffic scheduling and reducing the deployment complexity.**

***Index Terms*—Time-Sensitive Networking (TSN), Scheduling, ATS, P4**

I. Introduction

Time-Sensitive Networking (TSN) is a collection of standard extensions for the IEEE 802.1Q [1] Ethernet standard. TSN introduces multiple mechanisms that enable guaranteed QoS based on standard Ethernet. Different TSN mechanisms are able to offer different kinds of guarantees, ranging from token bucket traffic shaping to deterministic scheduled traffic with zero jitter. Without TSN mechanisms, the sole use of basic static priority mechanisms is known to suffer from starvation problems for the best-effort traffic class. For example, the Credit-based Shaper (CBS) restricts the flow rate, and the Time-Aware Shaper (TAS) may close higher priority queues for a certain time period [2].

TSN mechanisms are deployed on network switches, where scheduling and shaping decisions are made according to the used QoS mechanism and its configuration for egress ports. To integrate known TSN schedulers into a switch, the egress port scheduler needs to be capable of the intended mechanism at the design phase or capable of such a configuration later.

The domain-specific language P4 [3] enables the programmability of the data plane behavior on nowadays available network devices. However, P4 is constrained in its ability to program the behavior of a device's egress queue scheduling. Nevertheless, P4 devices can be programmed to run or emulate some stateful scheduling techniques, including variations of the priority queuing PIFO-mechanism [4] or Active Queue Management (AQM) algorithms [5].

[§]Equal contribution

A. Problem Statement

Typically, TSN configuration deployments are considered static. The network is not dynamically adjusting to current load situations. While critical traffic flow behavior and requirements may change over time, the deployed configuration does not adapt automatically. Furthermore, mechanisms with static resource reservations waste resources unnecessarily when flows change dynamically. We seek to find a resource-sharing mechanism that can be deployed on off-the-shelf programmable network devices like P4 switches.

In this paper, we analyze the Residence Delay Aggregation (RDA) approach, specifically designed for implementation on P4 switches. Unlike the traditional Least Slack Time First (LSTF) scheduling strategy, which prioritizes packets based on the remaining time to their deadline and necessitates a priority queue, RDA provides a solution for real-time dynamic scheduling on current P4 switches. Importantly, RDA dynamically determines when critical traffic should be prioritized.

II. Residence Delay Aggregation

The core concept of RDA, similar to LSTF, is that the priorities of deadline-carrying packets are determined by the total residence delay of packets as they travel through the network. The residence delay represents the cumulative time that a packet spends on switches while traversing the network. In a nutshell, packets with deadlines specified by the application are given a time allowance, encoded in the header of each packet, during which they can traverse between switches in best-effort queues (BEQ) without special handling. Upon the arrival of a deadline-carrying packet in a switch, the allowance time is checked before enqueuing the packet. If the allowance time does not suffice to meet the packet's deadline, the packet will instead be enqueued at a high-priority urgent queue (UQ).

A. Network Delays

TSN mechanisms achieve deterministic communication by providing upper bounds on the end-to-end delay. Figure 1 illustrates a simple network with two P4 programmable switches. The sender generates a packet (p) which is delivered to the receiver through switches sw_1 and sw_2. In addition to the link propagation delay (T_{prop}), each switch comprises a processing delay (T_{proc}) which we describe here as

$$T_{\text{proc}} = t_{\text{parser}} + t_{\text{ingress}} + t_q + t_{\text{egress}} + t_{\text{deparser}} \quad (1)$$

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.

©IEEE, (2024). *RDA: Residence Delay Aggregation for Time-Sensitive Networking* This is the author's version of the work. To appear in the Proceedings of IEEE/IFIP Network Operations and Management Symposium (NOMS) 2024.

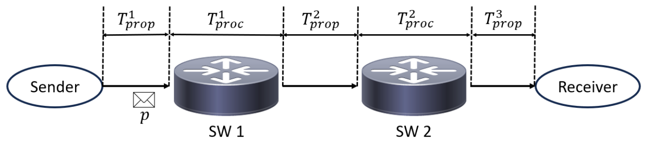

Fig. 1: Time-consumption along the transmission path. T_{prop} indicates propagation delay. T_{proc} indicates processing time within the switch.

In this paper, we let T_{proc} comprise the internal operations taking place within the programmable switches such as queuing and pipeline operations, as illustrated in Figure 2. Further elaboration of this figure is presented in Section II-B.

Upon the arrival of a packet in a switch, it is initially parsed by the parser and subsequently proceeds through the ingress pipeline until reaching the traffic manager. The time consumed by parsing and ingress processing is denoted by t_{parser} and t_{ingress}, respectively. The packet is then enqueued in the traffic manager and waiting to be dequeued. This queuing delay, denoted by t_q, depends on the total amount of traffic pushed in the queues and on the applied traffic shaping and scheduling algorithms. After the packet is dequeued from the traffic manager, it proceeds through the egress pipeline (assuming the switch offers an egress pipeline for packet processing) and the deparser to reach the egress port. The time consumed by the egress pipeline and deparser is denoted by t_{egress} and t_{deparser}, respectively. We assume that the factors t_{parser}, t_{ingress}, t_{egress}, and t_{deparser} are almost constant. Therefore, t_q stands as the sole factor that can affect T_{proc} and, consequently, impact the overall end-to-end latency.

B. Queuing Model

The existing TSN mechanisms prioritize traffic based on their types. According to the Class of Service in IEEE 802.1 Q standard, a 3-bit field Priority Code Point is presented in the VLAN header to indicate eight different classes of traffic. Therefore, a minimum of eight queues associated with fixed priority levels for each egress port are needed to isolate distinct traffic if the switch is required to support the complete set of traffic classes. Strict priority transmission selection ensures that higher-priority traffic is dequeued before lower-priority traffic, hence, the time-critical sessions running on the end devices can be accomplished in a timely manner.

One goal of TSN is to guarantee that packets reach their destination before a deadline. The urgency of reaching the deadline is the most critical factor for prioritizing traffic. The urgent traffic, which has an imminent deadline, should be assigned with higher priority than the non-urgent traffic, which still has a loose gap until the deadline.

In RDA, each switch dynamically sets traffic priorities based on two factors: the remaining time to meet the deadline and the current fill level of queues in the switch. Traffic may become more or less urgent in subsequent switches, depending on the queuing delay in the current switch. This dynamic priority assignment enables balancing traffic load while providing time guarantees for deadline-carrying packets.

RDA uses two separate queues for each egress port to buffer three types of traffic, as shown in the traffic manager in Fig. 2. An urgent queue (UQ) assigned with high priority is designated for buffering the urgent traffic, whereas a best-effort queue (BEQ) assigned with low priority is used for buffering the best-effort and non-urgent traffic. With strict priority transmission selection, the urgent traffic in the UQ is ensured to be transmitted before any traffic in the BEQ. A meter (also called policer) is applied for UQ to limit the urgent traffic rate. The meter has the burst and rate parameters $(B_{\text{meter}}, r_{\text{meter}})$ that denote the maximum burst size and the maximum long-term rate allowed for the UQ flows, respectively [6].

Note that, similar to existing TSN mechanisms, the deadline for packets can be guaranteed when the packet passes through the UQs of all switches along the path. In order to determine whether a packet is urgent or non-urgent, the switch should answer the question: Can the deadline of the incoming packet still be guaranteed when buffering the packet in the BEQ of this switch and in the UQs of all subsequent switches? Therefore, RDA prescribes that the packet header of each deadline-carrying packet contains a field called *time allowance* denoted as A, which specifies how long this packet can be buffered in the BEQ over the transmission path for switches $i \in \{1, ..., k\}$. The i-th switch is denoted by sw_i. The sender sets the initial value of A as

$$A = D - \sum_l T^l_{\text{prop}} - \sum_i T^i_{\text{pipe}} - \sum_i d^i_{UQ,\text{max}} \qquad (2)$$

where D is the deadline for reaching the destination. Here, l indicates the index of links along the transmission path; T^i_{pipe} represents the time consumed in sw_i except for the queuing delay in the traffic manager. The variable

$$d^i_{UQ,\text{max}} = \frac{B^i_{UQ,\text{meter}}}{r^i_{\text{line}}} \qquad (3)$$

for sw_i indicates the maximum queuing delay for the UQ, i.e., the maximum time until packets are served when assigned to this queue. $B^i_{UQ,\text{meter}}$ is the burst size for the meter assigned to the UQ in sw_i. The line rate for the egress port in sw_i is given by r^i_{line}. The maximum queuing delay for transmitting packets exclusively through the UQ in all switches is denoted as $\sum_i d^i_{UQ,\text{max}}$. This value, propagation and pipeline delays are then subtracted from the given deadline to establish the allowance time A for buffering in the BEQ.

The initial value of A is given by the sender, or the network controller which needs awareness of the delay values (2) along the path. A is initialized to be ≥ 0 to ensure on-time delivery because worst-case delays for subsequent UQs are accounted for during the initialization of A. Specifically, if no time remains for enqueuing the packet into BEQs, UQs are exclusively used to ensure an on-time packet delivery.

Note that, first, this proposal is conservative in the sense that we assume worst-case delays at every UQ. Secondly, with this method, we need to adjust A with the actual queuing delay at each switch after the packet is dequeued to reflect the actual

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.

©IEEE, (2024). *RDA: Residence Delay Aggregation for Time-Sensitive Networking* This is the author's version of the work. To appear in the Proceedings of IEEE/IFIP Network Operations and Management Symposium (NOMS) 2024.

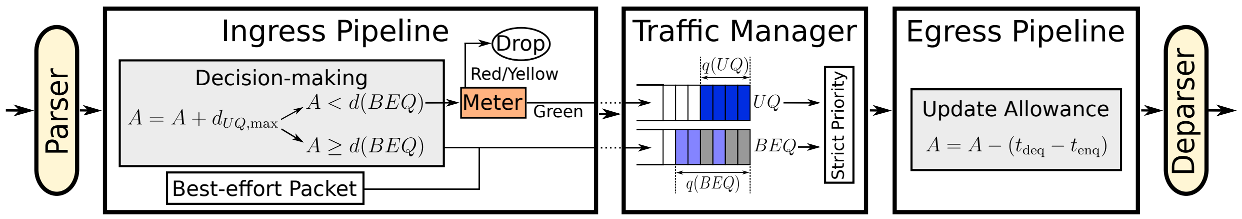

Fig. 2: System design of RDA. Packets in traffic manager: Navy blue: urgent traffic, Light blue: non-urgent traffic, Grey: best-effort traffic. t_{enq}: timestamp when packet is enqueued. t_{deq}: timestamp when packet is dequeued.

residence time. For an initial value of $A < 0$, the packet still may arrive within its deadline due to not fully utilized UQs along the path; however, no guarantees can be given.

Fig. 2 depicts the control flow when a packet (p) enters the switch. If p is a deadline-carrying packet observed by the parser, the decision-making process in the ingress pipeline is triggered. The value of A is first increased by $d^i_{UQ,\max}$ of this switch sw_i. This adjustment allows reclaiming the subtracted maximum queuing delay in the UQ from the packet deadline, as the packet has not yet entered the UQ at sw_i. This ensures the accurate representation of the packet's deadline constraints as it moves through the network.

The switch then compares the value of A with the threshold $d^i(BEQ)$ to decide which queue is eligible for p. The value of $d^i(BEQ)$ represents the queuing delay if p is buffered into the BEQ. This delay encompasses the transmission time of the current packets in the BEQ and UQ, as well as the upper bound delay caused by any packets that will enter the UQ in the future before p is eventually dequeued. Therefore, the classical delay bound $d^i(BEQ)$ at sw_i is given by

$$d^i(BEQ) = \frac{q^i(BEQ)}{r^i_{\text{line}} - r^i_{UQ,\text{meter}}} + \frac{B^i_{UQ,\text{meter}}}{r^i_{\text{line}} - r^i_{UQ,\text{meter}}} \tag{4}$$

where $q^i(BEQ)$ is the current depth of the BEQ, observed by an arriving packet. Should a packet be in risk of being dropped while being enqueued into the BEQ when BEQ is full, $d^i(BEQ)$ is assigned a value of ∞, to force the deadline-carrying packet into the UQ instead. The second term denotes an upper bound on the busy period of the UQ. This is a worst-case bound, as it assumes that the urgent packets are always enqueued into UQ at the meter rate ($r^i_{UQ,\text{meter}}$). The value of this upper bound is determined leveraging network calculus theory [6], as shown in Fig. 3. The arrival curve for UQ is expressed by affine function $\gamma_{r,b}(t) = rt + b$, where $r = r^i_{UQ,\text{meter}}$ and $b = B^i_{UQ,\text{meter}}$. The service curve for UQ is represented by peak rate function $\lambda_R(t) = Rt$, where $R = r_{line}$. The busy period of UQ lasts until the backlog reaches 0, which is the value of the second term in (4).

If $A \geq d^i(BEQ)$, then the packet is eligible to be enqueued in the BEQ, as the expected waiting time would not exceed the packet's deadline at sw_i. Otherwise, the UQ is selected. Recall that the UQ should be metered to ensure an upper bound of $d^i(BEQ)$. The packet p is dropped if it is set to be enqueued into UQ and UQ is currently full. This dropping mechanism ensures that the packets transferred to UQ see at

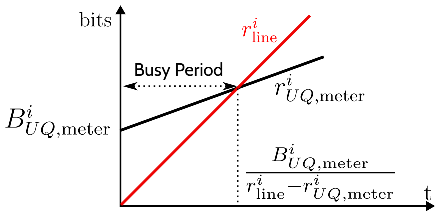

Fig. 3: The arrival and service curves of UQ. The arrival curve is colored in black, while the service curve is in red.

most the delay described in (3); however, such packet dropping should never appear under normal circumstances. The meter can also be applied to different traffic sources (e.g., based on flows) to prevent UQ from being flooded by a specific source with urgent traffic. Finally, after being deqeued, the value of A in the packet header is decreased with the **residence delay** ($t^i_{\text{deq}} - t^i_{\text{enq}}$) in the Egress pipeline of sw_i. The updated value of A is utilized at the next switch to decide which queue would be used for buffering this packet.

C. Limitations of Programmable Switches

The computation of $d^i(BEQ)$ requires knowing the current depth of the BEQ in the ingress pipeline, which is not supported by all programmable switch architectures (e.g., Tofino). The Tofino2 switch architecture allows a ghost thread to retrieve queue depths from the traffic manager to the ingress pipeline [7]. To accommodate different programmable switch architectures, the threshold $d^i(BEQ) = d^i_{BEQ,\max}$ (compare (4)) can be modified to the static

$$d^i_{BEQ,\max} = \frac{q^i_{BEQ,\max} + B^i_{UQ,\text{meter}}}{r^i_{\text{line}} - r^i_{UQ,\text{meter}}} \tag{5}$$

where $q^i_{\text{BEQ, max}}$ is the maximum depth of the BEQ at sw_i.

Note that the P4 language lacks support for division operations [5]. Hence, one approach to approximate the division operation is through performing bit shifting which requires the denominator to be a power of 2 for exact results [8]. To this end, the meter rate needs to be configured to a value that ensures ($r^i_{\text{line}} - r^i_{UQ,\text{meter}}$) is a power of 2.

D. Implement RDA in P4 Programmable Switches

The implementation of RDA relies on the programmable switch architecture, as described in Section II-C. The pseudocodes interpreted in this section are based on the bmv2 switch, which is a programmable software switch and does not support retrieving the current queue depth in the ingress

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.

©IEEE, (2024). *RDA: Residence Delay Aggregation for Time-Sensitive Networking* This is the author's version of the work. To appear in the Proceedings of IEEE/IFIP Network Operations and Management Symposium (NOMS) 2024.

```
if (meta.is_deadline_packet == 1) {
   meter_uq.execute_meter(0, meta.meter_tag);
   if (meta.meter_tag == 0) {
      hdr.rda.at = hdr.rda.at + meta.uq_max_del;
      q_dep = meta.beq_max_dep + meta.uq_bs;
      del_thr = q_dep >> meta.rate_diff_power;
      if (hdr.rda.at < del_thr) {
         standard_metadata.priority = 1;
      } else {
         standard_metadata.priority = 0;
      }
   } else {
      drop();
   }
} else {
   standard_metadata.priority = 0;
}
```

Listing 1: P4 pseudocode for decision-making, meter usage and queue assignment.

pipeline. Therefore, equation (5) is leveraged as the delay threshold for the decision-making procedure.

The P4 pseudocode for the ingress processing is presented in Listing 1. The values of $d^i_{UQ,\text{max}}$, $q^i_{BEQ,\text{max}}$, $B^i_{UQ,\text{meter}}$, and the exponent n in (6) are stored in the metadata of the programmable switch by `uq_max_del`, `beq_max_dep`, `uq_bs`, and `rate_diff_power`, respectively. These values are configured by the controller through the match-action table. Upon the arrival of a deadline-carrying packet, which is determined by the tag `is_deadline_packet`, the meter for constraining the rate of entering UQ is triggered. Note, that there is only one meter applied to UQ itself, not to each individual flow. The execution of meter results in three colors, as defined in RFC 2697 [9]. The `meter_tag` stores the result of the meter execution (according to the meter rate $r^i_{UQ,\text{meter}}$). A green result is indicated by 0. Once the deadline-carrying packet is allowed to be enqueued in UQ, the value of $d^i_{UQ,\text{max}}$ is reclaimed to the allowance time carried in the packet header. The value of delay threshold $d^i_{BEQ,\text{max}}$ is computed by shifting `q_dep` to the right by n bits, computed in:

$$2^n = r^i_{\text{line}} - r^i_{UQ,\text{meter}} \tag{6}$$

The arrived deadline-carrying packet is assigned a queue ID of 1 if the allowance time is less than the delay threshold. A higher queue ID corresponds to a higher queue priority. Thus, the queue with ID 1 indicates the UQ, whereas the BEQ is represented by the queue with ID 0.

Updating the allowance time in the egress pipeline is shown in Listing 2. The field `deq_timedelta` in `standard_metadata` stores the value of the time consumed by queuing in the traffic manager, which is then deducted from the current value of the allowance time.

III. Related Work

We briefly discuss related concepts in the following. The Push-In-First-Out (PIFO) [10] priority queuing concept is a promising approach to enable a flexible configuration of egress port schedulers. It is capable of emulating schedulers like Earliest Deadline First (EDF), Least Slack Time First (LSTF), and TSN Time-Aware Shaper (TAS) [11]. However, support for PIFO is unavailable on off-the-shelf devices and FPGAs are needed. Similarly, for other queueing challenges the use of FPGAs has proven to be a practical solution [12].

```
if (meta.is_deadline_packet == 1) {
   hdr.rda.at = hdr.rda.at -
        standard_metadata.deq_timedelta;
}
```

Listing 2: P4 pseudocode for updating the allowance time.

We also want to highlight one particular TSN mechanism due to similar goals: Asynchronous Traffic Shaping (ATS) as given by [13], [14]. Both RDA and ATS aim to provide end-to-end delay bounds per flow without reservations but achieve this through differing mechanisms. ATS employs multiple queues for each egress port to separate ingress flows based on ingress port and priority. Each queue calculates an eligibility time for its topmost flow, retaining packets until this time is met. Unlike RDA, there is no need for additional encoded information (allowance time) in packets during transmission. However, ATS necessitates a more complex hardware support including queuing structure and queuing scheduling, leading to increased resource overhead compared to RDA. Currently, no off-the-shelf devices are available that support ATS with the required queuing and scheduling strategies. In contrast, RDA only requires priority queues and a strict priority transmission selection algorithm, which is specified as the default algorithm for selecting frame for transmission in IEEE 802.1Q [1]. RDA is implementable on Tofino-based switches solely with the availability of strict priority scheduling and FIFO-queues.

IV. Conclusion

We proposed a new dynamic scheduling mechanism called Residence Delay Aggregation (RDA), which is designed to provide time guarantees for time-critical traffic while only utilizing priority queuing when necessary to deliver packets timely. The presented approach is built on top of dynamic traffic scheduling at each switch, prioritizing traffic based on the urgency of meeting packet deadlines and the current queue states. RDA is designed to be implementable on the modern P4 programmable switches. In future work, we plan to assess the performance of RDA across various hardware platforms in different network topologies and traffic scenarios. The impact on the performance due to the limitation imposed by the P4 programming language and the underlying concepts will be investigated. Additionally, we will extend RDA by incorporating multiple urgent queues to enhance its flexibility in prioritization.

Acknowledgement

This work is funded by the Federal Ministry for Economics and Climate Action (BMWK) within the joint research project Reallabor Antrieb 4.0, and by the German Research Foundation (DFG) as part of projects B4, T3 within the Collaborative Research Center (CRC) 1053 – MAKI.

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.

©IEEE, (2024). *RDA: Residence Delay Aggregation for Time-Sensitive Networking* This is the author's version of the work. To appear in the Proceedings of IEEE/IFIP Network Operations and Management Symposium (NOMS) 2024.

References

[1] "Ieee standard for local and metropolitan area networks–bridges and bridged networks," *IEEE Std 802.1Q-2022 (Revision of IEEE Std 802.1Q-2018)*, pp. 1–2163, 2022.

[2] J. Falk, D. Hellmanns, B. Carabelli, N. Nayak, F. Dürr, S. Kehrer, and K. Rothermel, "Nesting: Simulating ieee time-sensitive networking (tsn) in omnet++," in *2019 International Conference on Networked Systems (NetSys)*. IEEE, 2019, pp. 1–8.

[3] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: Programming protocol-independent packet processors," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, p. 87–95, jul 2014. [Online]. Available: https://doi.org/10.1145/2656877.2656890

[4] A. G. Alcoz, A. Dietmüller, and L. Vanbever, "Sp-pifo: Approximating push-in-first-out behaviors using strict-priority queues," in *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, 2020, pp. 59–76.

[5] R. Kundel, A. Rizk, J. Blendin, B. Koldehofe, R. Hark, and R. Steinmetz, "P4-codel: Experiences on programmable data plane hardware," in *ICC 2021 - IEEE International Conference on Communications*, 2021, pp. 1–6.

[6] J.-Y. Le Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer Berlin Heidelberg, 2001.

[7] A. Agrawal and C. Kim, "Intel tofino2 – a 12.9tbps p4-programmable ethernet switch," in *2020 IEEE Hot Chips 32 Symposium (HCS)*, 2020, pp. 1–32.

[8] P. Golchin, C. Zhou, P. Agnihotri, M. Hajizadeh, R. Kundel, and R. Steinmetz, "Cml-ids: Enhancing intrusion detection in sdn through collaborative machine learning," in *2023 19th International Conference on Network and Service Management (CNSM)*. IEEE, 2023, pp. 1–9.

[9] D. J. Heinanen and D. R. Guerin, "A Single Rate Three Color Marker," RFC 2697, Sep. 1999. [Online]. Available: https://www.rfc-editor.org/info/rfc2697

[10] A. Sivaraman, S. Subramanian, M. Alizadeh, S. Chole, S.-T. Chuang, A. Agrawal, H. Balakrishnan, T. Edsall, S. Katti, and N. McKeown, "Programmable packet scheduling at line rate," in *Proceedings of the 2016 ACM SIGCOMM Conference*, ser. SIGCOMM '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 44–57. [Online]. Available: https://doi.org/10.1145/2934872.2934899

[11] C. Gärtner, A. Rizk, B. Koldehofe, R. Hark, R. Guillaume, R. Kundel, and R. Steinmetz, "Poster: Leveraging pifo queues for scheduling in time-sensitive networks," in *2021 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, 2021, pp. 1–2.

[12] R. Kundel, L. Nobach, H.-J. Kolbe, T. Meuser, and R. Steinmetz, "Fpga-assisted massive packet queueing and traffic shaping at the network edge," in *2022 IEEE 30th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2022, pp. 1–1.

[13] "Ieee standard for local and metropolitan area networks–bridges and bridged networks - amendment 34: Asynchronous traffic shaping," *IEEE Std 802.1Qcr-2020 (Amendment to IEEE Std 802.1Q-2018 as amended by IEEE Std 802.1Qcp-2018, IEEE Std 802.1Qcc-2018, IEEE Std 802.1Qcy-2019, and IEEE Std 802.1Qcx-2020)*, pp. 1–151, 2020.

[14] J. Specht and S. Samii, "Urgency-based scheduler for time-sensitive switched ethernet networks," in *2016 28th Euromicro Conference on Real-Time Systems (ECRTS)*, 2016, pp. 75–85.

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.