Constant Delay Switching: Asynchronous Traffic Shaping with Jitter Control

Alexej Grigorjew*, Florian Metzger*, Tobias Hoßfeld*, Johannes Specht†,
Franz-Josef Götz‡, Feng Chen‡, Jürgen Schmitt‡
*University of Würzburg, Germany – {alexej.grigorjew | florian.metzger | tobias.hossfeld}@uni-wuerzburg.de
†University of Duisburg-Essen, Germany – johannes.specht@uni-due.de
‡Siemens AG, Germany – {franz-josef.goetz | chen.feng | juergen.jues.schmitt}@siemens.com

***Abstract*—Recent developments in the Time Sensitive Networking (TSN) group have put forth a number of mechanisms to improve deterministic latency in layer 2 switches. Their shaping approaches differ with respect to prerequisites, flexibility, maximum latency, and latency variation (jitter). Asynchronous solutions provide flexibility in dynamic scenarios, but Credit-Based Shaping (CBS) and Asynchronous Traffic Shaping (ATS) come with a high latency variance, as the minimum latency is not affected by them. Some applications in industrial and automotive use cases require low jitter while trying to avoid a dependency on time synchronization, and even TSN's redundancy mechanism benefits from a lower latency variance.**

This work is bridging the gap between low-jitter transmission requirements and asynchronous shaping mechanisms. In this paper, the application of a stateless Asynchronous Constant Delay Shaper (ACDS) is discussed. It collects the queuing duration from the previous bridge to delay each frame for the remaining time such that a constant per-hop delay for each frame is achieved. The discussion includes the prerequisites, the mechanism itself, and its technical complexity, as well as a formal analysis of its guaranteed latency by proving its relation to a token-bucket based ATS. Further, two extensions are discussed, and an evaluation shows the applicability of ACDS by comparing its jitter to ATS in a frame-level simulation.

I. Introduction

While technological innovations bring advancements into many fields of our lives, such as industrial automation and transportation, the underlying requirements to their communication infrastructure are also increasing. A certain amount of predictability is often required by interconnected systems, in other words, a deterministic behavior of the underlying communication system is presumed. Traditionally, these requirements were often met by various bus systems such as PROFIBUS and CAN bus, partly due to their simple and cheap application. Recent advancements have increased the bandwidth requirements of these systems, which can no longer be met by traditional buses. As a result, Ethernet-based systems have been adapted to be used in time-sensitive scenarios to provide similar guarantees, which led to many different solutions, such as PROFINET, EtherCAT, and AFDX.

In order to support the development and interoperability of predictable forwarding solutions, the IEEE Time-Sensitive Networking (TSN) [1] task group is developing a common set of standards that extend layer 2 Ethernet switches by additional Quality of Service (QoS) capabilities for predictable networking. These standards implement a wide set of features that have previously been fragmented and used independently by the different deterministic Ethernet solutions. As a result, multiple mechanisms are available with vastly different methods of operation, features, requirements, flexibility, and accuracy. Most notably, concepts for bounded worst-case latency can be split into synchronized approaches (Time-Aware Shaping [2], Cyclic Queuing and Forwarding [3]) and asynchronous approaches (Credit-Based Shaping [4], Asynchronous Traffic Shaping [5], [6]).

It is often preferred to avoid the requirement for time synchronization, so the former approaches cannot be used there. In addition, it is often easier to implement dynamic plug-and-play stream reservation with asynchronous approaches. However, while the current asynchronous mechanisms help to achieve a bounded maximum latency, they still leave the minimum latency unattended which can result in a large delay variance (jitter). High jitter cannot be tolerated by some time-sensitive applications due to limited queuing capabilities, and even the network itself can benefit from lower delay variance. For example, jitter could lead to frame reordering with Frame Replication and Elimination [7] when losses on one replicated path occur, and higher jitter causes a higher resource utilization during the elimination phase, because transmitted frames must be remembered for a longer time in order to eliminate their replicated copy. Further, while Asynchronous Traffic Shaping (ATS) simplifies per-hop latency computation during stream reservation, it relies on a per-stream token bucket state at each hop, which can be undesirable when looking at core switches in large networks with many streams.

Contribution: In this work, these two problems are addressed while remaining interoperable with the regular ATS mechanism. A traffic damping mechanism, Asynchronous Constant Delay Shaper (ACDS), is proposed that works in a stateless manner by sharing frame delay information between bridges. The queuing delay information is inserted after the Ethernet header, and extracted by the subsequent switch to achieve a constant per-hop delay. This document presents the requirements and implications of ACDS. It proves that the damper's formal latency bound is equal to the upper bound given by ATS. In addition, head of line blocking in shared queues is analyzed and it is shown that ACDS can be used

ISBN 978-3-903176-48-5

Table I
USED VARIABLES, SYMBOLS, AND ABBREVIATIONS IN THIS DOCUMENT

Variable	Description
TQ	Transmission Queue
Prop	Refers to **Prop**agation delay
SF	Refers to Store and Forward delay (transmission delay)
Proc	Refers to **Proc**essing delay (switch fabric)
Shaper	Refers to Shaping delay (or damper delay)
d^{abc}	Delay of the process *abc* (being one of the above)
$d^{abc,def}$	Combined delay of *abc* and *def* (e.g., $d^{TQ,SF,Shaper}$)
d_i^{abc}	Delay component *abc* as perceived by stream i
f_i	A particular frame of stream i
$\mathcal{H}_i$	Set of all streams with a higher priority than i
$\mathcal{E}_i$	Set of all streams with the same priority as i
$\mathcal{L}_i$	Set of all streams with a lower priority than i
$\mathcal{Q}_i$	Set of all streams with the same priority as i that share the same shaping queue (i.e., same ingress port as i)
p	A certain, fixed priority level
p_i	The priority level of frames f_i from stream i
δ_p	Per-hop delay guarantee of priority p at the current link
$\delta_p(h)$	Per-hop delay guarantee of priority p at the link h
c	A latency class c. Each priority p can be split into multiple classes c.
c_i	The latency class of frames f_i from stream i
b_i	Burst size of stream i; upper bound on the short-term amount of data
r_i	Leak rate of stream i; upper bound on the long-term data rate
$\hat{\ell}_i$	Maximum length of frames from stream i
$\check{\ell}_i$	Minimum length of frames from stream i
r	Link speed (e.g., $1\,\mathrm{Gbit/s}$)
t_1 .. t_7	Local bridge time at certain points in the bridge pipeline
$\Delta t_{a,b}$	Time interval between t_a and t_b: $\Delta t_{a,b} = t_b - t_a$.
$w_i(\Delta t)$	Cumulative amount of packet data (number of bits) that was transmitted by stream i in the time interval Δt
$t_1(f_i)$	Local bridge time t_1 as perceived by the frame f_i, i.e., the local time at which f_i is enqueued into the transmission queue

with the same queue allocation rules as ATS [6]. Finally, an evaluation is presented that compares the jitter of ACDS and ATS by means of example scenarios in a simulation.

The remainder of this document is structured as follows. Section II presents the background information that is necessary for the analysis, including the switch delay model (II-B), related work (II-C), and the existing ATS and its traffic model (II-D). In Section III, the proposed new ACDS mechanism is explained in full detail. This includes prerequisites (III-A), the applied shaping function (III-B), an analysis of queue allocation rules (III-C), the resulting delay guarantee (III-D), and two possible extensions for more flexibility (III-E, III-F). Section IV presents the evaluation, Section V contains a short discussion, and Section VI concludes the paper.

II. BACKGROUND AND RELATED WORK

All used symbols in this document are explained in Table I. Thereby, sets are always uppercase calligraphic letters ($\mathcal{H}_i$, $\mathcal{E}_i$, $\mathcal{L}_i$ and $\mathcal{Q}_i$). Small letters represent an element of a set ($x \in \mathcal{Q}_i$) or individual values (link speed r, burst size b_i). Any properties of the directly *observed* stream are referred to with index i, while any other *interfering* streams are referred to with index x.

A. Priority Levels and Queues

Each frame belongs to a certain priority level which can be mapped to a specific traffic class by the switches. Note that this mapping is not necessarily equal in every switch in the network. This document assumes priority transmission as in IEEE standard 802.1Q [8], up to eight traffic classes are supported at the egress, and each class has their own first-in first-out (FIFO) transmission queue.

This document introduces traffic damping via ACDS as an additional shaper in the switch pipeline. In this model, the shaper is applied before the frame enters the transmission queue, and each shaper comes with its own shaper queue(s). The concrete number of queues is addressed by the queue allocation rules in Section III-C. In general, for each priority level and each ingress port, an individual shaper queue is required at each egress port.

Note that this two-level queuing model is used to describe the behavior of the switch. It is possible to achieve the same behavior with a single-level queue implementation by applying more sophisticated transmission selection algorithms.

B. Switch Delay Model

Figure 1 presents an abstract model of the transmission process from one bridge to another, including a shaping mechanism before priority transmission selection. In this work, the latency of a transmission is regarded from shaper to shaper, as opposed to measuring the time from in-port to in-port (or egress to egress). This model is in-line with the per-hop latency analysis of ATS [6] and is preferred because the situation starts with a well-defined traffic model after the shaper.

The latency is measured right after the frame f_i is considered eligible by the shaper (circle in Fig. 1) at the moment t_1. It is enqueued in the transmission queue (TQ) of its traffic class at the respective egress port. At the moment t_2, the frame f_i is picked by the priority transmission selection mechanism. The time spent by f_i in the transmission queue is given by $d^{TQ} = \Delta t_{1,2} = t_2 - t_1$. The frame is transmitted towards the next hop. At the moment t_3, the first bit is received after the propagation delay, and t_4 marks the reception of the last bit. The transmission delay (store and forward, SF) of the frame can be expressed by $d^{SF} = \Delta t_{3,4} = t_4 - t_3$. Similarly, the internal processing time of the switch fabric (ASIC, CPU, TCAM, ...) is given by $d^{Proc} = \Delta t_{4,5} = t_5 - t_4$.

Finally, at t_5 the frame is enqueued into its respective shaper queue. This queue operates in FIFO order, the number of individual queues is addressed later in Section III-C. Note that the shaper only processes the head of the shaper queue at any given time. If subsequent frames in the shaper queue are eligible for transmission, they will only be processed after the current head of the queue is released. The moment when a frame f_i arrives at the head of its shaper queue is denoted by t_6. The moment when the shaper flags the head frame f_i as eligible for transmission and hands it to the transmission queue (TQ) is denoted by t_7. The entire shaping delay can be written as $d^{Shaper} = \Delta t_{5,7} = t_7 - t_5$. The moment t_7 equals the start t_1 of the transmission towards the next hop (bridge 3).

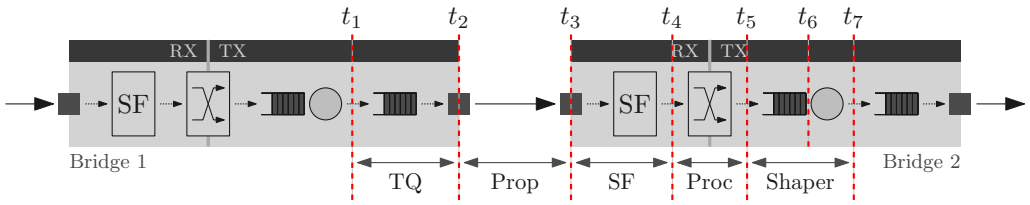

Figure 1. Illustration of per-hop delay components.

Note that the moments $t_{\{1..7\}}$ refer to local times in their respective bridges. Clocks are not synchronized in this model. In particular, the propagation delay d^{Prop} cannot be derived by $t_3 - t_2$, as these times are measured based on different reference systems. When minimizing the latency jitter of a transmission, the propagation delay is of less interest as all frames along the path experience the same propagation delay.

An alternative model may be considered in which transmission d^{SF} and processing d^{Proc} are interleaved instead of consecutive. For the application of the proposed damper, it is sufficient if $d^{SF,Proc} = \Delta t_{3,5} = t_5 - t_3$ holds. Cut-through switching is not considered here, as its benefits would be negated by ACDS anyways.

C. Related Literature and Concepts

1) Related work: This paper applies traffic damping to reduce delay jitter by examining the queuing delay of the previous bridge and adjusting the eligibility time of the current bridge's shaper accordingly. It is based on our previous technical report [9]. A very similar mechanism has been proposed later in [10], but the authors only briefly state that the latency model from ATS [6] can be applied, without formal proof. Queuing policies are mentioned less formally without proof, and practical implications in real-world ASICs are discussed instead. This work presents a more formal approach that mathematically shows that the latency bound and queue allocation rules actually perform as expected. Further, in [11], the authors analyze end-to-end delay implications of traffic dampers with more realistic, non-ideal clocks using Network Calculus. Their results require network-wide knowledge and increased computational effort, while our analysis is applicable to distributed scenarios and low-profile CPUs. Generally, traffic damping has been proposed as early as 1993 by Zhang and Ferrari [12], [13] under the name of *delay-jitter controlling regulator*. However, they do not formally address the problem of head of line blocking when using interleaved regulators (shared queues, only the head is examined by the shaper). Later in 1998, Cruz proposed a scheduling algorithm [14] that uses aggregation and statistical multiplexing for more efficient resource utilization. Their equations can be used during network planning in order to pre-configure and pre-compute *guaranteed virtual paths* such that the QoS of Best-Effort traffic remains acceptable. Once again, they assume network-wide knowledge for their computations, which is not always feasible in dynamic reservation use cases.

2) Similar TSN and DetNet mechanisms: Several standards developed under the TSN working group are also aiming towards eliminating jitter and providing low delay guarantees. Time-Aware Shaping (TAS, 802.1Qbv [2]) allows full isolation of streams and deterministic delays, but it requires network-wide time synchronization [15] and a central control unit with network-wide stream information for full jitter control. The proposed mechanism ACDS does not rely on synchronization and can be implemented in a distributed control plane without pre-computed gate control lists. A more dynamic approach is presented by Cyclic Queuing and Forwarding (CQF, 802.1Qch [3]) which uses TAS to implement a dual buffer mechanism which cycles between transmitting and receiving for both buffers. This can achieve jitter-free transmissions with reduced complexity, but it still relies on time synchronization, and all switches in the network are required to have the same cycle time. ACDS can be used with non-periodic traffic and can be configured differently on a per-hop basis, possibly increasing the available latency budget at bottlenecks, while lowering the latency budget at the network edge.

On layer 3, the DetNet group [16] aims to provide similar guarantees to the layer 2 TSN efforts. Most notably, the standards RSVP-IntServ (RFC 2210 [17]) and Guaranteed QoS (RFC 2212 [18]) specify a per-stream shaping mechanism and the latency math behind it. These specifications require per-stream buffering, and they only consider maximum latency, they do not attempt to minimize jitter. Nowadays, the DetNet group is collaborating more closely with the TSN group and relies on some of their lower layer specifications.

3) In-band Network Telemetry: The proposed traffic damper ACDS relies on delay information that is carried over by the previous bridge towards the shaping bridge. This is achieved by injecting the experienced queuing delay as an additional header field into the payload of the Ethernet frame, such that the subsequent bridge can extract it. The general idea to embed network telemetry into data plane frames without requiring intervention of the control plane is recently being discussed as *In-band Network Telemetry* (INT) [19]. Under this term, it is primarily being used in the context of Software-Defined Networks for monitoring and diagnosis purposes. With increasing popularity of programmable switches, most notably P4 [20], more advanced actions became available. In [21], the authors use in-frame delay information and link delay computations to meet QoS thresholds by dropping (or marking) frames that are likely to exceed their delay requirement anyways. Although they also share delay information in

their data plane traffic, they designed their system for applications that permit packet loss, while in TSN shaping is used and resources are reserved explicitly to prevent congestion-related packet loss entirely. At the current time, P4 cannot be used to fine-tune shaping delays as the scheduler remains a black box and is not programmable via P4.

D. Asynchronous Traffic Shaping

This work makes use of the specification and delay analysis of Asynchronous Traffic Shaping (ATS) [5], as previously published under the name Urgency-Based Scheduler [6]. This section covers the most important aspects of its analysis. In particular, the traffic model, the queue allocation rules, and the worst-case latency bound are of importance.

1) Traffic model: There is no requirement for time synchronization, each talker may emit its frames asynchronously. However, each flow f_i must satisfy a leaky bucket constraint [22], [23] characterized by a burstiness b_i and a leak rate r_i. For any arbitrary time duration Δt, the leaky bucket constraint limits the cumulative packet data $w_i(\Delta t)$ of flow f_i by:

$$w_i(\Delta t) \leq b_i + \Delta t \cdot r_i. \tag{1}$$

Many more specific traffic models can be mapped to this general model. For example, periodic traffic with a fixed packet inter-arrival time T_i and a fixed packet length ℓ_i could be described by $r_i = \frac{\ell_i}{T_i}$ and $b_i = \ell_i$.

The applied traffic shaper re-enforces the constraint in Equation 1 at every bridge such that the requirements for the ATS delay bound are satisfied at each hop on a frame's path.

2) Queue allocation rules: In order to prevent head of line blocking in the shaper queue that could otherwise violate the latency bound, ATS applies three queue allocation rules (QARs) that regulate which frames may share the same shaper queue [6].

QAR 1: Frames f_i and f_x are not allowed to share a shaper queue if both are received from different servers, i.e., if both are received by different in-ports.

QAR 2: Frames f_i and f_x are not allowed to share a queue if both are sent in different priority levels to the current bridge.

QAR 3: Frames f_i and f_x are not allowed to share a queue if both are sent in different priority levels from the current bridge.

In particular, if both f_i and f_x are received by the same ingress port and belong to the same priority level, they are allowed to share the same shaper queue.

3) ATS latency bound: If the above conditions are met and Asynchronous Traffic Shaping is used, an upper bound for the queuing (TQ), transmission (SF), and shaper latency can be derived. The maximum per-hop latency $d_i^{TQ,SF,Shaper}$ of all frames f_i from a flow i is bounded by [6, Eq. 21]

$$d_i^{TQ,SF,Shaper} \leq \max_{x \in \mathcal{Q}_i} \left(\frac{b_{\mathcal{H}_i} + b_{\mathcal{E}_i} - \check{\ell}_x + \hat{\ell}_{\mathcal{L}_i}}{r - r_{\mathcal{H}_i}} + \frac{\check{\ell}_x}{r} \right). \tag{2}$$

Thereby, $\mathcal{Q}_i$ is a subset of all streams $\mathcal{E}_i$ with the same priority as i that are received on the same ingress port and therefore share a queue with f_i. $b_{\mathcal{H}_i} = \sum_{x \in \mathcal{H}_i} b_x$ is the sum of all bursts of streams x with higher priority p_x than i, $b_{\mathcal{E}_i} = \sum_{x \in \mathcal{E}_i} b_x$ is the sum of all bursts from equal priority streams (including i itself), $\hat{\ell}_{\mathcal{L}_i} = \max_{x \in \mathcal{L}_i}(\hat{\ell}_x)$ is the largest maximum frame size $\hat{\ell}_x$ from all lower priority streams x, and $r_{\mathcal{H}_i} = \sum_{x \in \mathcal{H}_i} r_x$ is the sum of all leaky bucket rates from higher priority streams. Equation 2 can be simplified if all involved streams are assumed to have the same (worst case) minimum frame size $\check{\ell} = 64\,\text{bytes}$:

$$d_i^{TQ,SF,Shaper} \leq \frac{b_{\mathcal{H}_i} + b_{\mathcal{E}_i} - \check{\ell} + \hat{\ell}_{\mathcal{L}_i}}{r - r_{\mathcal{H}_i}} + \frac{\check{\ell}}{r}. \tag{3}$$

Equation 3 returns the same for all streams of the same priority level, as the computation of the maximum and all individual differences are vanished by this simplification. The impact on the accuracy is minimal as the formula is quickly dominated by the sum of bursts $b_{\mathcal{H}_i} + b_{\mathcal{E}_i}$ and rates $r_{\mathcal{H}_i}$ [6].

III. Combining ATS with Jitter Control

The original idea of Asynchronous Traffic Shaping (ATS) is to re-shape the traffic at every hop in order to refresh the leaky bucket constraint from Equation 1. For that, only the inter-arrival times between two frames are relevant in order to prevent accumulating bursts on a path. The individual delay of a single frame is not directly considered in the shaping process. This leads to high variance in the end-to-end delays of individual frames, i.e., high jitter, especially under low load conditions when the buckets within the shapers can recover frequently. The minimum possible delay and the worst case delay may be far apart.

Therefore, the application of *traffic damping* [24] is suggested in order to re-shape every frame individually to its maximum delay, effectively minimizing jitter. The remainder of this section covers the application of Asynchronous Constant Delay Shaping (ACDS) in more detail, along with its prerequisites and resulting guarantees.

A. Assumptions and Prerequisites

The application of traffic dampers comes with three major conditions.

1) The local time intervals $\Delta t_{1,2}$ and $\Delta t_{3,6}$ can be measured and are available to the data plane (i.e., the shaper) of the bridge.

2) The bridge is able to push additional fields to the frame's header at the moment t_2, i.e., after the frame is selected for transmission and dequeued from the transmission queue. In particular, the local value $\Delta t_{1,2}$ of bridge 1 must be transmitted to bridge 2.

3) All steps in the depicted pipeline of Figure 1 preserve the frames' order, i.e., no reordering can occur for two frames that share the same queue.

4) Further, the accuracy of the technique can be improved if the bridge is able to assess the propagation delay $d^{Prop} = \Delta t_{2,3}$ of a particular link. This would improve the accuracy of the end-to-end delay itself, but it is not strictly required for jitter control since (i) the propagation delay of a particular link is constant, and (ii) all frames of the same stream traverse the

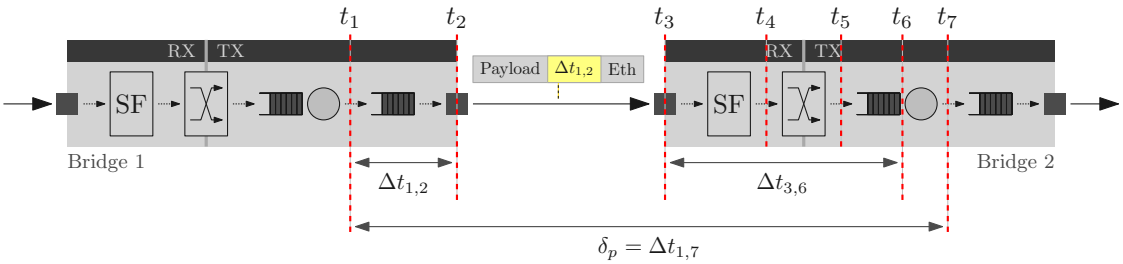

Figure 2. The transmission queue delay $\Delta t_{1,2}$ of bridge 1 is shared with bridge 2. $\Delta t_{6,7}$ compensates any jitter in the previous delay components.

same links. Without loss of generality, the remainder of this work assumes that the propagation delay $d^{Prop} = \Delta t_{2,3}$ is known to the receiving bridge of a transmission.

B. Specification of Shaping Delay

Let δ_p be the per-hop delay guarantee configured for traffic class p on the observed switch. If the above prerequisites are met, the eligibility time t_7 of the ACDS can be specified by:

$$t_7 = t_1 + \delta_p \tag{4}$$

This is illustrated in Figure 2. Further, the individual shaping delay $\Delta t_{6,7}$ of the shaper queue's head frame in bridge 2 is specified by:

$$\begin{aligned} \Delta t_{6,7} &= t_7 - t_6 \\ &= t_1 + \delta_p - t_6 \\ &= t_1 + \delta_p - (t_1 + \Delta t_{1,2} + \Delta t_{2,3} + \Delta t_{3,6}) \\ &= \delta_p - \Delta t_{1,2} - \Delta t_{2,3} - \Delta t_{3,6} \end{aligned} \tag{5}$$

The per-hop delay configuration δ_p and the time intervals $\Delta t_{2,3}$ and $\Delta t_{3,6}$ are known to the receiving bridge 2 as per the above assumptions (Section III-A). The propagation delay $\Delta t_{2,3}$ can be omitted with a minor impact on accuracy of delay upper bounds, as mentioned earlier. The transmission queue delay $\Delta t_{1,2}$ is transmitted in an extra header field by the previous bridge 1, as illustrated by Figure 2.

This methodology does not require the bridges to hold any per-stream state beyond the lifetime of each individual frame. However, it depends on the information $\Delta t_{1,2}$ received by the previous bridge. Some implications of this method are briefly discussed in Section V.

C. Avoiding Head of Line Blocking

Assume that the frames in the shaper queue are numbered according to their order, where f_1 represents the head of the queue. The shaping delay $\Delta t_{6,7}$ is only applied to the head f_1 of the shaper queue. Therefore, if the subsequent frames $\{f_2, f_3, ...\}$ require a smaller shaping delay, they exceed their eligibility time in the shaper queue, i.e., they are subject to head of line (HoL) blocking.

Similarly to ATS, the impact of HoL blocking on the latency bounds can be avoided by employing additional shaper queues and preventing certain frames from sharing the same queue. The number of required queues and their queue allocation rules (QARs) are discussed in the following.

Head of line blocking is avoided if the eligibility time $t_7(f_2)$ of any subsequent frame f_2 is always later than the eligibility time $t_7(f_1)$ of the current head frame f_1:

$$\begin{aligned} & t_7(f_2) \geq t_7(f_1) \\ \Leftrightarrow\ & t_1(f_2) + \delta_{p_2} \geq t_1(f_1) + \delta_{p_1} \end{aligned} \tag{6}$$

Equation 6 can be simplified if both frames belong to the same priority, i.e., if $\delta_{p_2} = \delta_{p_1}$. This can be achieved by separating frames with different traffic classes into different shaper queues (QAR 2 and 3). All frames in the same queue have the same delay configuration δ_p and Equation 6 can be simplified:

$$t_1(f_2) \geq t_1(f_1) \tag{7}$$

To satisfy the remaining condition, refer to Figure 3. The bridges A and B send frames towards the same bridge C, which will be transmitted further by the same egress port of C. W.l.o.g., assume that $t_1^A(f_1) < t_1^B(f_2) = t_1^A(f_1) + \Delta t_{A,B}$, and assume that the transmission queue delay $d^{TQ} = \Delta t_{1,2}$ of bridge A is longer than d^{TQ} of B by $\Delta t_{A,B}$:

$$d_A^{TQ}(f_1) > d_B^{TQ}(f_2) + \Delta t_{A,B} \tag{8}$$

If all other delays (propagation, transmission, processing) are similar and both frames share the same shaper queue, frame f_2 arrives in the queue before f_1 does. However, due to the smaller queuing delay $\Delta t_{1,2}^B = d_B^{TQ}(f_2)$, the ACDS damper assigns a higher shaping delay to f_2 to compensate it, effectively blocking f_1 which would be eligible for transmission earlier. Therefore, frames arriving from different ingress ports may not share the same shaper queue (QAR 1).

If all frames that share the same shaper queue arrive from the same in-port (bridge A), it always holds

$$t_1^A(f_2) \geq t_1^A(f_1) \ \Leftrightarrow\ t_1^C(f_2) \geq t_1^C(f_1) \tag{9}$$

since the individual steps of the bridge pipeline preserve frame order as per Assumption 3.

In summary, head of line blocking can be prevented by applying the same queue allocation rules as ATS:

QAR 1: Frames f_i and f_x are not allowed to share a shaper queue if both are received from different servers, i.e., if both are received by different in-ports in the current bridge (due to Equation 7).

QAR 2: Frames f_i and f_x are not allowed to share a queue if both are *received* in different priority levels ($p_i \neq p_x$) by

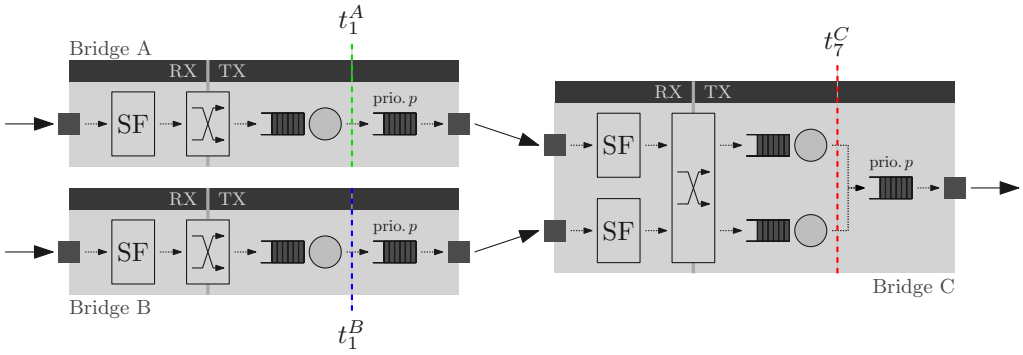

Figure 3. Frame isolation in different shaper queues to avoid head of line blocking.

the current bridge (due to Equation 7, as this would also imply that they came from different transmission queues).

QAR 3: Frames f_i and f_x are not allowed to share a queue if both are *sent* in different priority levels ($p_i \neq p_x$) by the current bridge (due to Equation 6).

D. Possible Per-hop Delay Configurations

Naturally, Equation 4 only holds if the per-hop delay configuration is always larger than the delay itself:

$$\delta_p \geq \Delta t_{1,7} \tag{10}$$

During the network configuration phase, a per-hop delay configuration δ_p must be found for each port and each traffic class p such that the worst-case latency is always below δ_p. Further, during network operation, a formal latency model is required that is used during admission control to verify that the worst-case delay is actually below δ_p in the current situation.

This section shows that the delay upper bound from ATS [6] can also be applied to ACDS. The analysis of the ATS latency relies on two key properties: (i) the queue allocation rules, and (ii) the traffic specifications of the streams coming out of each shaper (at t_1). As shown in Section III-C, ACDS is subject to the same queue allocation rules as ATS. If the traffic specification in Section II-D1 can also be satisfied at every bridge on a frame's path, the ATS latency bound from Equations 2 and 3 can be applied.

Assume that the talkers emit their frames in accordance to the leaky bucket constraint (cf. Section II-D1), i.e., they limit their cumulative packet data $w_i(\Delta t_{a,b})$ during any arbitrary duration $\Delta t_{a,b} = t_b - t_a$ to

$$w_i(\Delta t_{a,b}) \leq b_i + r_i \cdot \Delta t_{a,b}. \tag{11}$$

Further, assume that the constraint is violated at the n-th hop along the stream's path, i.e., $w_i(\Delta t_{x,y}) > b_i + r_i \cdot \Delta t_{x,y}$. Traffic damping enforces a constant per-hop delay $d^{const} = \sum_{k=1}^{n} \delta_p(h_k)$ to all frames of a stream, i.e., if f_1 is transmitted at the moment t_a from the talker, it is marked eligible at the moment $t_a + d^{const}$ by the shaper at the n-th hop. If the constraint is violated, it implies that either there have been unexpected sources of delay that can jitter, or that the talker initially sent more traffic than specified by Equation 11. This implies that the talker transmitted more data than $b_i + r_i \cdot \Delta t_{x-d^{const},y-d^{const}}$ in the time interval $[t_x - d^{const}, t_y - d^{const}]$. This is a contradiction to the initial assumption from Equation 11, because $\Delta t_{x-d^{const},y-d^{const}} = (t_y - d^{const}) - (t_x - d^{const}) = t_y - t_x = \Delta t_{x,y}$, which is an instance of the arbitrary duration $\Delta t_{a,b}$.

This means that the ATS leaky bucket constraint holds for ACDS if the talkers initially emit their traffic under the same constraint. Then the per-hop delay of ACDS-shaped streams is constrained by the ATS latency bound:

$$d_i^{TQ,SF,Shaper} \leq \frac{b_{\mathcal{H}_i} + b_{\mathcal{E}_i} - \check{\ell} + \hat{\ell}_{\mathcal{L}_i}}{r - r_{\mathcal{H}_i}} + \frac{\check{\ell}}{r} \leq \delta_{p_i}. \tag{12}$$

In summary, the traffic damping mechanism ACDS presented in this document is (i) subject to the same queue allocation rules as ATS, (ii) keeps the leaky bucket condition enforced if it is initially maintained by the talkers, and therefore, (iii) it can provide the same per-hop latency guarantees as ATS. It does not require to maintain per-stream state in the data plane of the bridges and keeps the end-to-end delay jitter at a minimum due to constant delays.

E. Increasing the Number of Per-hop Delay Configurations

Up to this point, there has been exactly one delay configuration δ_p for each link and each traffic class p. In general, each traffic class could even support more than one per-hop delay. This could be useful in FRER [7] environments to equalize the cumulative delays of two (or more) redundant paths with different lengths. Generally, it would allow more flexibility for streams with similar priority but vastly different latency requirements, and it extends the number of configurations beyond the maximum number of supported traffic classes.

To support multiple per-hop delay configurations, each traffic class p could be split into multiple *latency sub-classes* c. The latency of stream i would then be shaped to the value of δ_{c_i} instead of the global guarantee δ_{p_i}. For each such per-hop delay δ_{c_i}, a separate shaper queue is required. An alternative set of queue allocation rules could be formulated as follows:

QAR 1: Frames f_i and f_x are not allowed to share a shaper queue if both are received from different servers, i.e., if both are received by different in-ports in the current bridge.

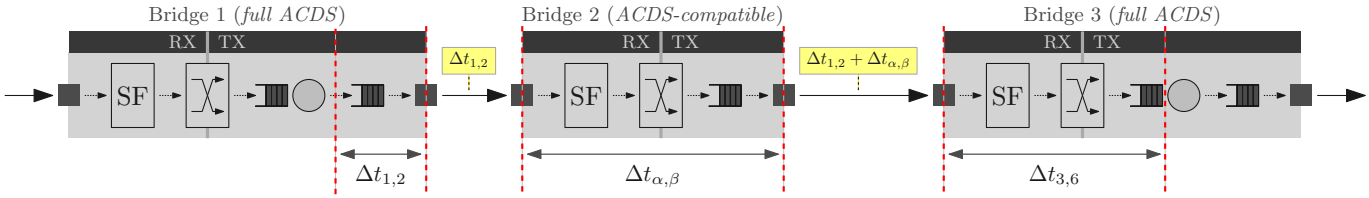

Figure 4. Extension of jitter compensation to *ACDS-compatible* switches that are not capable of shaping themselves.

QAR 2**b**: Frames f_i and f_x are not allowed to share a queue if both are received in different *latency classes* ($c_i \neq c_x$) by the current bridge.

QAR 3**b**: Frames f_i and f_x are not allowed to share a queue if both are sent in different *latency classes* ($c_i \neq c_x$) by the current bridge.

Without loss of generality, the remainder of this document assumes a single latency class per traffic class p and a single delay configuration δ_p for each traffic class.

F. Extending Jitter Control Beyond ACDS Switches

Given the methodology from Section III-B, it is only possible to compensate jitter from transmission queues of a single link if both devices of that link support ACDS. If an intermediate device does not support ACDS, it introduces jitter to the end-to-end latency which will not be compensated by later ACDS-capable devices. But if this intermediate device can still *measure* its own ingress-to-egress delay and push that information to the Ethernet header (without shaping), this information could be used by the next full ACDS device to compensate for its jitter later. Switches that cannot shape their traffic but can still measure and push their latency will be referred to as *ACDS-compatible* devices.

Figure 4 demonstrates this extension with a single ACDS-compatible switch between two full ACDS switches. The entire ingress-to-egress delay of bridge 2 ($\Delta t_{\alpha,\beta}$), including the additional store and forward latency, is added to the existing delay information $\Delta t_{1,2}$.

This extension is particularly useful for switches capable of in-band network telemetry, as they are currently popular in programmable data plane communities, such as P4 switches (Sec. II-C3). These devices typically lack sophisticated QoS features, but they can be freely programmed to measure and insert delay information. The accuracy can vary depending on the programmed target; therefore ASIC-based implementations, such as Intel's Tofino, are recommended for this.

Note that this extension requires that QAR 1 is extended to isolate frames from different paths into different queues. Only separating them by their ingress port is not sufficient anymore. The full path of a frame is network-wide information that cannot easily be obtained by bridge 3 alone. Therefore, such an extension requires more effort on the control plane (path-aware reservation protocol, or external controllers with an overview of the paths). These approaches are out of scope for this paper.

IV. Evaluation

The evaluation is split into two parts: worst-case comparison and statistical jitter. In the worst-case comparison, the theoretical highest possible jitter is calculated for ATS and ACDS for a given topology. For the statistical jitter, a frame-level discrete event simulation has been conducted with SimPy[1].

A. Topology Setup

Figure 5 shows the topologies (A and B) of the simulated scenarios. The talker (end device) on the very left is communicating with the listener on the very right. The delays of the red stream are observed. In total, there are 7 bridges on its path, and from each bridge, 14 additional streams are transmitted over the same path (blue), which adds up to 99 streams sharing the last link. In topology A, each of the talkers is communicating with the same listener. In topology B, the last link from bridge 7 to the listener is split into two links, such that the observed (red) stream is separated from the rest.

Each stream has a fixed frame size of 250 bytes (258 bytes including the preamble, and 270 bytes also including the inter-frame gap). The frames' inter-arrival times are uniformly distributed between 240 µs and 260 µs, representing periodic streams with slight deviation. This leads to a 85% bandwidth utilization at the last link. The processing delays of all bridges are uniformly distributed between 1 µs and 5 µs. The propagation delay was omitted.

B. Worst-case Comparison

In the context of deterministic networking, the performance of a mechanism is better described by its worst-case guaranteed bounds than by measurements and statistical characteristics. This section includes a short analysis of worst-case jitter with ATS and ACDS for the two example topologies.

In both topologies, the first transmission from talker to switch is assumed to have no jitter in this evaluation, as there are no interfering streams. Afterwards, ACDS ensures zero jitter between the 7 bridges, while for ATS, the possible jitter is given by the difference between the minimum and maximum latency. The minimum end-to-end delay up to bridge 7 is given by seven transmission delays with no queuing and the minimum of 1 µs of processing delay: $\min(d_{1,7}^{ATS}) = 7 \cdot \left(\frac{258\,\text{byte}}{1\,\text{Gbit/s}} + 1\,\mu\text{s}\right) = 21.5\,\mu\text{s}$. The maximum end-to-end delay is given by the sum of Equation 3 for each of the seven bridges, plus the worst-case of 5 µs switch fabric delay: $\max(d_{1,7}^{ATS}) =$

[1] https://simpy.readthedocs.io/

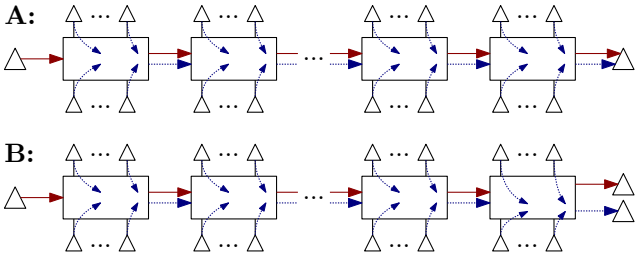

Figure 5. Linear topologies (A and B) used for jitter evaluation. 7 bridges with 14 additional talkers per bridge (blue streams) are deployed.

664.1 µs. Up to bridge 7, ATS has already accumulated a total worst-case jitter of $\max(d_{1,7}^{ATS}) - \min(d_{1,7}^{ATS}) = 642.6\,\mu s$, while for ACDS, the jitter is still zero for both topologies.

The last hop differs between the topologies, but it behaves similarly for both shapers, because the listener does not actually reshape incoming traffic. In topology A, all 99 streams interfere with each other, leading to a maximum queuing delay of $\max(d_{7,8}^{TQ}) = 98 \cdot \frac{270\,\text{byte}}{1\,\text{Gbit/s}} = 211.7\,\mu s$ and $\min(d_{7,8}^{TQ}) = 0$. In topology B, the observed stream is separated from the others in the last transmission queue. The only remaining jitter could be caused by the store and forward operation if the frame lengths were variable (which they are not in this scenario). Hence, the jitter from the last hop (bridge 7 to listener) is zero for both shapers in topology B.

In summary, the worst-case end-to-end jitter of ATS is 854.3 µs (topology A) and 642.6 µs (topology B), respectively, while the worst-case jitter of ACDS is 211.7 µs (topology A) and 0 µs (topology B), respectively. Note that in reality, it is expected that all links are also populated by lower priority Best-Effort transmissions. This potentially increases the jitter on the last (unshaped) hop by one MTU transmission time: $\frac{1542\,\text{byte}}{1\,\text{Gbit/s}} = 12.3\,\mu s$. Further, note that this analysis requires knowledge of all currently active streams in the network. In a distributed control plane, a bridge would pessimistically resort to $\max(d_{1,7}^{ATS}) = 7 \cdot \delta_p = 1750\,\mu s$, as more detailed information of the other streams is not available to it.

C. Measured Jitter During Simulation

In the area of deterministic networking, where guarantees, proven bounds, and 0% losses are desired, a statistical measurement only has a limited informative value of the underlying performance. Nevertheless, measurements and simulations are still useful to validate a new mechanism, and they help to get a better understanding of it. Further, both ACDS and ATS could theoretically be used outside of a deterministic environment, and knowing their *expected* influence on the traffic, as opposed to a (possibly extremely unlikely) guarantee, can be a valuable KPI for such use cases.

In order to increase the observed jitter artificially and to reduce simulation time, the talkers start to transmit their frames at a similar time, and drift apart during the simulation due to random inaccuracies (inter-arrival times vary between 240 µs and 260 µs). In addition, all talkers skip every 5-th frame in their periodic schedule. Missing periodic frames may occur commonly in a dynamic environment, for example

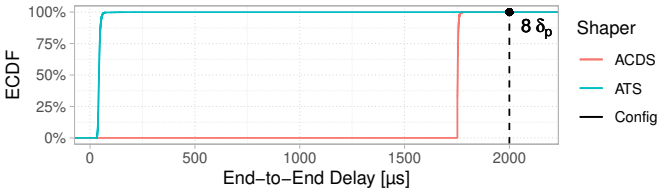

Figure 6. Comparison of delay distributions for ATS and ACDS. Only topology A is shown, as both scenarios behave similarly.

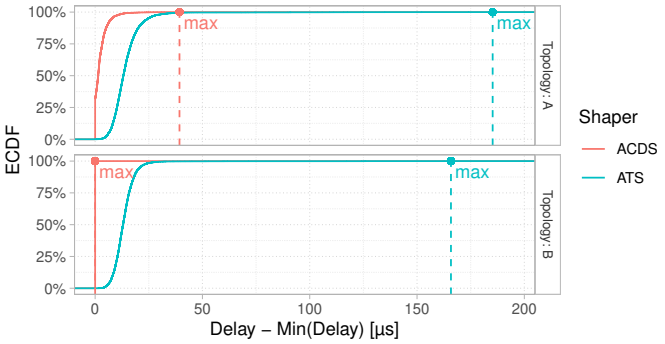

Figure 7. Comparison of jitter distributions for ATS and ACDS in a 3 seconds simulation.

due to reboots, short outages, changes in the topology, or reconfigurations of the bridges.

Figure 6 shows the empirical cumulative distribution functions (ECDF) of the observed end-to-end delay values for the observed stream. Only topology A is shown, as both of them behave very similarly. The colors indicate the shaping mechanism. The dashed black line indicates the end-to-end delay configuration as a reference: $8\,\delta_p = 8 \cdot 250\,\mu s = 2\,ms$.

As expected, the delay of ACDS is much higher (mean 1.75 ms) compared to ATS (mean 43 µs). All frames of ACDS on the first 7 hops are equalized to their pre-configured value $\delta_p = 250\,\mu s$. The theoretical worst-case latency of both mechanisms is only 879.9 µs, which is also extremely unlikely to be seen in practice without artificially forcing it by controlling every single frame transmission directly. The lower values of ATS show the multiplexing gain in a randomly distributed superposition of multiple periodic transmission cycles. The variance of the ATS curve appears slightly bigger, but this must be reflected at a smaller scale.

Figure 7 shows the delay distributions normalized by their minimum value, i.e., each observed end-to-end delay is reduced by the minimum observed value for that shaper, so both curves start at 0. This reduces the size of the x-axis and shows the differences in more detail. In addition, the highest observed value for each shaper is marked by a vertical dashed line of the same color.

As previously discussed, the observed jitter of ACDS is zero for topology B. As for topology A, ACDS shows a steep increase with more than 25% of all frames observed having the same minimum end-to-end delay (1752 µs). The maximum observed jitter is also much lower at 39.3 µs compared to the 185 µs of ATS. The remaining jitter in the red ACDS

curve of Figure 7 for topology A is caused by the variance in the queuing delay at the last hop, as discussed earlier. In summary, the effective jitter of the observed scenario could be reduced significantly by applying ACDS, and some effects of the previous worst-case analysis could be confirmed.

V. Discussion

This section briefly discusses two challenges that could arise when implementing this mechanism in practice. First, it is often mentioned that even hardware switches are subject to inaccuracies with respect to clock drift, time stamping inaccuracies, and sequential handling of simultaneous events by the switch fabric [10], [11]. Inaccuracies are generally an issue to be aware of, but hardware time stamping with deviations as small as $8\,\mathrm{ns}$ is already common in networking hardware. Further, while sequential handling of frames with equal eligibility times is dangerous in this regard, it is entirely mitigated by using the *computed* eligibility time for t_1 (or t_7) in Figure 2 rather than the *measured* time of the frame's extraction from the shaper queue. That way, possible inaccuracies presented by simultaneous events are compensated by the next shaper, just like the queuing delay jitter.

Further, the delay information that is pushed into the Ethernet frame is subject to transmission errors, just as any other information in the frame. The checksum in the Ethernet header must be updated at every hop to be adjusted to the new Ethernet payload. This has not been necessary before, and just like any other function in the switch, it presents another risk of failure in a deterministic environment. In order to reduce the impact of this (and other failures) significantly, asymmetric signatures can be applied instead of symmetric CRCs to protect against hardware faults of a single bridge [24].

VI. Conclusion

This work presented Asynchronous Constant Delay Shaping (ACDS), a traffic damping mechanism that achieves the same latency guarantee as Asynchronous Traffic Shaping (ATS) without the requirement for per-stream state and with a very tight jitter control. It proved that the underlying queue allocation rules of ACDS are the same as those of ATS and presented two possible extensions for a more flexible implementation. The evaluation showed that the jitter can be reduced significantly for an example topology from industrial automation, both by analyzing the worst-case jitter, and by measuring the actual jitter by means of a frame-level simulation.

Further, the discussion includes possible difficulties when transferring the theoretical model into practice. Future work in this subject may focus on (i) clarifying and mitigating possible inaccuracies of time stamping and switch fabric, (ii) on improving frame integrity, for example by applying asymmetric signatures to detect and protect against hardware faults, and (iii) on implementing this approach in a real switch.

Acknowledgement

This research was partly funded by the Bavarian Ministry of Economic Affairs, Regional Development and Energy under grant number DIK0250/02, project KOSINU5.

References

[1] IEEE, *Time-Sensitive Networking (TSN) Task Group*. [Online]. Available: https://1.ieee802.org/tsn/.

[2] "IEEE Standard for Local and Metropolitan Area Network – Bridges and Bridged Networks – Amendment 25: Enhancements for Scheduled Traffic," *IEEE Std 802.1Qbv*, 2015.

[3] "IEEE Standard for Local and Metropolitan Area Networks — Bridges and Bridged Networks — Amendment: Cyclic Queuing and Forwarding," *IEEE Std 802.1Qch*, 2017.

[4] "IEEE Standard for Local and Metropolitan Area Networks – Virtual Bridged Local Area Networks – Amendment: Forwarding and Queuing Enhancements for Time-Sensitive Streams," *IEEE Std 802.1Qav*, 2010.

[5] "Draft Standard for Local and Metropolitan Area Networks — Bridges and Bridged Networks — Amendment: Asynchronous Traffic Shaping," *IEEE P802.1Qcr/D2.0*, 2019.

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

[7] "Standard for Local and Metropolitan Area Networks – Frame Replication and Elimination for Reliability," *IEEE Std 802.1CB*, 2017.

[8] "IEEE Standard for Local and Metropolitan Area Network – Bridges and Bridged Networks," *IEEE Std 802.1Q*, 2018.

[9] A. Grigorjew, F. Metzger, T. Hoßfeld, *et al.*, "Asynchronous traffic shaping with jitter control," University of Wuerzburg, Institut für Informatik, Tech. Rep., Jun. 2020.

[10] T. Eckert, A. Clemm, and S. Bryant, "Glbf: Per-flow stateless packet forwarding with guaranteed latency and near-synchronous jitter," in *2021 17th International Conference on Network and Service Management (CNSM)*, IEEE, 2021, pp. 578–584.

[11] E. Mohammadpour and J.-Y. L. Boudec, "Analysis of dampers in time-sensitive networks with non-ideal clocks," *arXiv preprint arXiv:2109.02757*, 2021.

[12] H. Zhang and D. Ferrari, "Rate-controlled static-priority queueing," in *IEEE INFOCOM'93 The Conference on Computer Communications, Proceedings*, IEEE, 1993, pp. 227–236.

[13] H. Zhang, "Service disciplines for guaranteed performance service in packet-switching networks," *Proceedings of the IEEE*, vol. 83, no. 10, pp. 1374–1396, 1995.

[14] R. L. Cruz, "Sced+: Efficient management of quality of service guarantees," in *Proceedings. IEEE INFOCOM'98, the Conference on Computer Communications. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Gateway to the 21st Century (Cat. No. 98*, IEEE, vol. 2, 1998, pp. 625–634.

[15] "IEEE Standard for Local and Metropolitan Area Networks – Timing and Synchronization for Time-Sensitive Applications," *IEEE P802.1AS-Rev*, 2019.

[16] IETF, *Deterministic Networking (detnet) Working Group*. [Online]. Available: https://datatracker.ietf.org/wg/detnet/about/.

[17] J. T. Wroclawski, *The Use of RSVP with IETF Integrated Services*, RFC 2210, Sep. 1997. DOI: 10.17487/RFC2210. [Online]. Available: https://rfc-editor.org/rfc/rfc2210.txt.

[18] S. Shenker, C. Partridge, and R. Guerin, *Specification of Guaranteed Quality of Service*, RFC 2212, Sep. 1997. DOI: 10.17487/RFC2212. [Online]. Available: https://rfc-editor.org/rfc/rfc2212.txt.

[19] C. Kim, A. Sivaraman, N. Katta, A. Bas, A. Dixit, and L. J. Wobker, "In-band network telemetry via programmable dataplanes," in *ACM SIGCOMM*, 2015.

[20] P. Bosshart, D. Daly, G. Gibb, *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[21] B. Turkovic, S. Biswal, A. Vijay, A. Hüfner, and F. Kuipers, "P4qos: Qos-based packet processing with p4," in *IEEE 7th International Conference on Network Softwarization (NetSoft)*, IEEE, 2021, pp. 216–220.

[22] R. L. Cruz, "A calculus for network delay. i. network elements in isolation," *IEEE Transactions on information theory*, vol. 37, no. 1, pp. 114–131, 1991.

[23] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The single-node case," *IEEE/ACM transactions on networking*, vol. 1, no. 3, pp. 344–357, 1993.

[24] J. Specht, *Dampers with forward traffic isolation*, 2019. [Online]. Available: http://www.ieee802.org/1/files/public/docs2019/new-specht-dampers-fti-1119-v01.pdf (visited on 04/15/2020).