

# Decentralized Latency Models and Information Flow for TSN Shapers – A Tutorial

Alexej Grigorjew\*, Stefan Geissler\*, Jürgen Schmitt†, and Tobias Hossfeld\*, *Senior Member, IEEE*

**Abstract**—Recent advancements in the fields of industrial automation and in-vehicle communication have also increased the requirements on their underlying networks. Traditional field bus technologies are being replaced by regular Ethernet devices with features from the IEEE Time-Sensitive Networking working group. Their standards are designed to achieve zero congestion loss and deterministic latency bounds, but the specific guarantees can differ depending on the shapers used and how the bounds are calculated. After several contributions to the standardization efforts, this work presents a number of decentralized latency models that can be applied locally on the switches, without requiring global knowledge from the rest of the network. It includes a formal definition of delay segments that allows to incorporate interoperability between different shapers and latency models in the future. It presents pseudo-code suggestions for the latency bounds computation of Strict Priority, Credit-Based Shaping, Asynchronous Traffic Shaping, and Cyclic Queuing and Forwarding, all of which can be applied locally on the devices. Finally, five models were implemented and evaluated with respect to their efficiency in terms of latency, jitter, and number of accepted streams. The results illustrate how different shapers are suited for different networks and traffic distributions.

**Index Terms**—Admission control, decentralized, deterministic networking, latency bounds, real-time networking, resource reservation, shaper comparison, Time-Sensitive Networking, TSN

## I. INTRODUCTION

Real-time networking has recently become a major focus in scientific literature and standardization efforts. Real-time systems provide guaranteed transmissions with deterministic upper bounds on latency and jitter, which has been a critical requirement in industrial automation and in-vehicle communication for decades. Traditional field bus solutions providing deterministic guarantees are migrating to Ethernet-based networks to satisfy increasing bandwidth demands. However, these second-generation industrial Ethernet systems (e.g., EtherCAT, PROFINET) suffer from closed, proprietary, and homogeneous designs, making them incompatible and requiring gateways for inter-domain communication. Furthermore, unlike main Ethernet standards that scale with technology, proprietary systems often remain locked at their original transfer speeds.

To bridge these gaps, the third generation is being developed by the IEEE Time-Sensitive Networking (TSN) task group

within the Ethernet ecosystem. Instead of proprietary solutions, TSN introduces openly standardized tools for regular Ethernet switches to enable *converged networking*, allowing Best-Effort and real-time traffic to coexist on shared infrastructure. These mechanisms encompass time synchronization, various shapers, transparent redundancy, and diverse configuration protocols. However, these are individual tools; their correct combination, implementation, and management are left to vendors and operators. While standard profiles suggest feature subsets for specific use cases (e.g., industrial automation [1]), they lack full-fledged communication system implementations comparable to legacy technologies.

Often, these individual tools are orchestrated by a Central Network Controller (CNC) [2] for resource management and switch configuration. A core CNC task is resource reservation and admission control based on accurate worst-case latency bounds for reserved streams. While many publications address worst-case latency computation for various shapers [3][4] targeting CNC use cases, their models typically require global topology and stream information. Latency models for decentralized systems are rare, and papers addressing protocol adaptation [5] to acquire this information are scarcer. Nevertheless, decentralized latency models offer significant benefits. Crucially, a solid decentralized mathematical foundation can foster true interoperability, i.e., guaranteeing latency across disparate technologies, which is essential for fully converged networking.

Figure 1 illustrates three scenarios highlighting the benefits of decentralized models. First, Fig. 1a depicts a fully decentralized setup where each switch manages its own allocations (colored areas) via decentralized control logic. This avoids the CNC single point of failure and accommodates heterogeneous switch architectures, where a single CNC might have trouble if the internal implementations are different from what it expects. Conversely, Fig. 1b uses two CNCs for separate TSN domains. Each domain is configured homogeneously by its CNC but may use different mechanisms. Here, the inter-domain link can leverage decentralized latency models for end-to-end guarantees. Finally, Fig. 1c shows a FRER (Frame Replication and Elimination for Reliability) [6] example. The green switch on the left replicates traffic for redundancy, and the red switch on the right eliminates duplicates. However, the remaining traffic can still exceed original burstiness [6, Sec.C.9], doubling downstream resource demands. Asynchronous Traffic Shaping (ATS) on the red node can mitigate this, but it is rarely implemented due to complex queuing requirements. Transitioning from other shapers to ATS momentarily solves this issue cost-effectively.

This work was partly funded by the Bavarian Ministry of Economic Affairs, Regional Development and Energy under grant number DIK0250/02 (project KOSINUS), and by the Deutsche Forschungsgesellschaft DFG (German Research Agency) under grant number 316878574 (project SDN-App).

\*Chair of Communication Networks, University of Würzburg, Germany

†Siemens AG, Germany

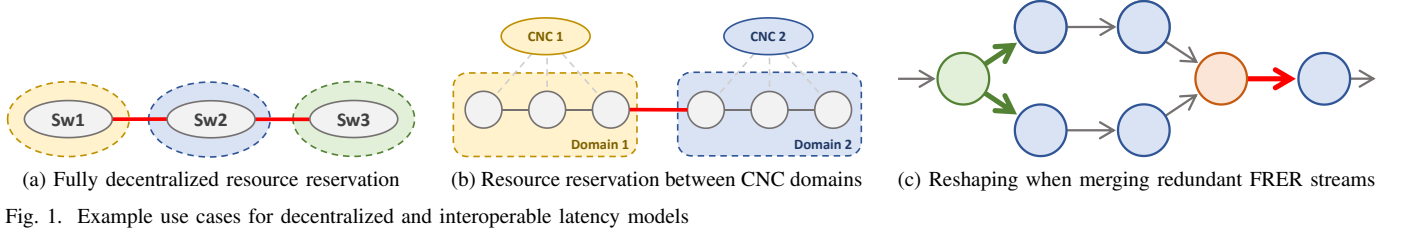


TABLE I  
NUMBER OF TAGGED PUBLICATIONS BY VENUE.

| # Pubs | Venue                                                                                |
|--------|--------------------------------------------------------------------------------------|
| 44     | IEEE Access                                                                          |
| 30     | IEEE Transactions on Industrial Informatics                                          |
| 10     | IEEE Transactions on Network and Service Management                                  |
| 5      | IEEE International Conference on Emerging Technologies and Factory Automation (ETFA) |
| 5      | IEEE/ACM Transactions on Networking                                                  |
| 4      | IEEE International Conference on Communications                                      |
| 4      | IEEE Transactions on Vehicular Technology                                            |
| 3      | IEEE Communications Surveys & Tutorials                                              |
| 3      | IEEE Global Communications Conference                                                |
| 3      | IEEE International Workshop on Factory Communication Systems (WFCS)                  |
| 40     | Other (< 3 papers each)                                                              |

Concurrently, the TSN group is developing the Resource Allocation Protocol (RAP) [7], a publish-subscribe reservation protocol for both centralized and decentralized systems. RAP announces streams, allows end devices to attach, and performs local latency calculations and configurations. To contribute to these efforts, this tutorial addresses the following **research questions**: (RQ1, Sec. VI) How can worst-case delay bounds be derived locally and decentralized, without global topology and stream knowledge? (RQ2, Sec. V) What information is required for local delay bound computation, and how is it acquired during reservation? (RQ3, Sec. IV&VIII) How can decentralized latency models facilitate shaper transitions in heterogeneous environments? (RQ4, Sec. VII) How can different latency models with varying assumptions be compared fairly without bias? (RQ5, Sec. VII) Which shaper is recommended for specific application scenarios?

The remainder of this work is structured as follows. Section II reviews related work and identifies the research gap. Section III introduces the notation, symbols, and terms. Section IV defines *delay segments*, which redefine the borders of responsibilities for switch, while Section V details the decentralized information flow. Section VI presents formal models and pseudo-code implementations for latency computation. Section VII evaluates their efficiency in an empirical reservation study. Finally, Section VIII provides a discussion and future outlook, and Section IX concludes the paper.

## II. OVERVIEW ON RELATED WORK

Recent literature tackles deterministic Ethernet latency from multiple directions. For systematic overviews, readers may refer to existing surveys. Specifically, [8] covers TSN standards and research up to 2018, while more recent surveys focus on distinct use cases like industrial automation [9], vehicular

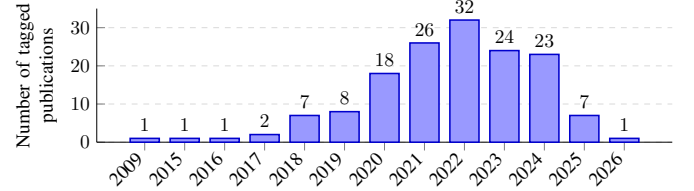


Fig. 2. Number of tagged publications by year.

networks [10], or TSN-5G integration [11]. Other surveys target specific methodologies, such as Time-Aware Shaping scheduling [12] or Network Calculus-based worst-case latency models [3].

Crucially, none of these surveys address decentralized applicability or the signaling details required to gather information. To map recent research trends, this section structurally overviews publications selected via relevant search terms, which were manually categorized. The initial list was built by querying the OpenAlex<sup>1</sup> database using search terms including *Time-Sensitive Networking*, *TSN*, *shaper*, *shaping*, *admission*, *scheduling*, *latency*, *cyclic*, *worst-case delay*, *Network Calculus*, *trajectory approach*, *queuing theory*, *delay bound*, and *decentralized*. Queries used various keyword combinations, each retrieving the top 60 most-cited results from the past decade. Merging results and removing duplicates yielded 311 publications. Each was manually reviewed and assigned the following tags: (i) **‘survey/tutorial’** for general overviews; (ii) **‘standards/technical’** for introducing standards or proposing related algorithmic advancements like new shapers; (iii) **‘5G/6G/wireless’** for combining TSN with wireless technologies. The tags (iv) **‘central’** and (v) **‘decentral’** specified the architecture. Specifically, ‘central’ was applied if a central authority (e.g., SDN controller, TSN CNC) or global topology knowledge was required. Conversely, ‘decentral’ was assigned only if the authors explicitly noted applicability to decentralized networks. We did not infer potential applicability if unstated. Papers matching neither were classified as ‘not mentioned’. Furthermore, (vi) **‘scheduling’** denoted Gate Control List (GCL) configurations; (vii) **‘routing’** indicated path-finding contributions; and (viii) **‘worst-case delay analysis’** marked papers analyzing deterministic upper delay bounds, including but not limited to (ix) **‘Network Calculus’**. Additionally, (x) **‘reservation’** signified explicit stream admission control rather than just pre-determined delay minimization, while (xi) **‘measurement’** required actual data packet transmissions in physical testbeds or simulations, ex-

<sup>1</sup><https://openalex.org/> is a bibliographic catalogue of scientific papers

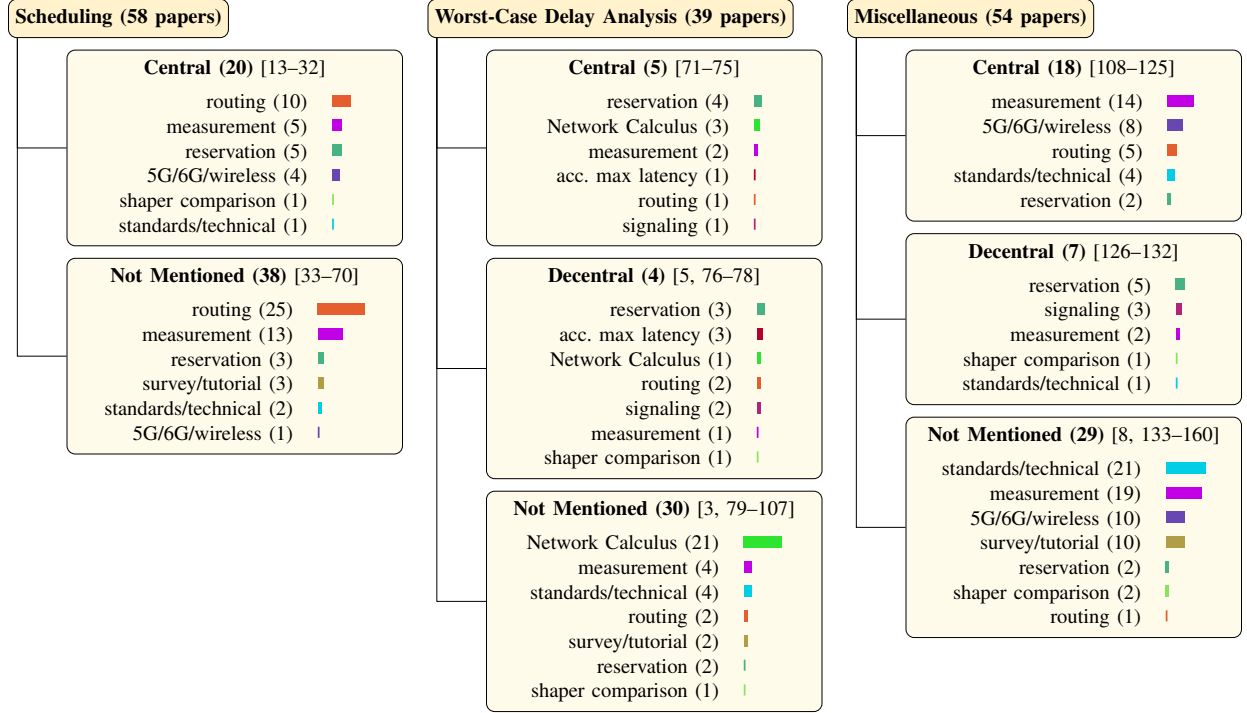


Fig. 3. Distribution of topics in related publications; (parentheses) indicate number of publications; [brackets] contain reference IDs

cluding purely algorithmic evaluations. Lastly, (xii) ‘**shaper comparison**’ indicated performance evaluations of multiple shaping strategies under identical scenarios; (xiii) ‘**signaling**’ marked papers describing the utilization or adaptation of reservation protocols; and (xiv) ‘**acc. max latency**’ specified explicit use of the *accumulated maximum latency* field in SRP or RAP for calculations. For context, this tutorial would be tagged with ‘survey/tutorial’, ‘decentral’, ‘worst-case delay analysis’, ‘Network Calculus’, ‘reservation’, ‘shaper comparison’, ‘signaling’, and ‘acc. max latency’.

From the initial 311 publications, 160 unrelated papers were excluded. Table I outlines the remaining publications by venue, which is incidentally dominated by IEEE journals, while Figure 2 illustrates their chronological distribution. Notably, over 50% of these papers were published between 2022–2026. Figure 3 details the final tag distribution, where papers are categorized under ‘scheduling’, ‘worst-case delay analysis’, or ‘miscellaneous’. These categories are further subdivided into ‘central’, ‘decentral’, and ‘not mentioned’. Each subgroup lists its respective references alongside the distribution of the 10 remaining tags.

The ‘scheduling’ category contains the majority of publications and lacks a ‘decentral’ subgroup. Its most common supplementary tags are ‘routing’ and ‘measurement’, reflecting the prevalent research focus: resolving joint scheduling and routing problems via GCL computation and validating the configurations empirically. Although 38 scheduling papers omit mentioning a central authority, most of them likely depend on it due to the nature of the scheduling problem: temporal isolation of streams on their end-to-end path. The next largest group comprises 54 ‘miscellaneous’ papers. As they are neither configuring GCL schedules nor computing

worst-case delays, they primarily address measurements and technical advancements like new shaping mechanisms and protocols. Within the ‘decentral’ subgroup, seven publications focus heavily on resource reservation and signaling protocols [126, 129], compare centralized and decentralized admission control [130–132], evaluate optimal resource configuration [127], or adapt GCL configurations dynamically [128]. The ‘worst-case delay analysis’ group is heavily dominated by ‘Network Calculus’ (NC) publications, which provide a convenient framework for deterministic latency calculations. While most delay analysis papers omit explicit mention of architectural applicability, their arrival curves typically require preceding shaper and path details, and their service curves rely on per-stream concatenation and leftover-service properties [3]. Consequently, these works implicitly assume a central controller unless local information acquisition is explicitly addressed. Furthermore, most shapers, excluding Asynchronous Traffic Shaping (ATS), suffer from hop-by-hop burstiness cascading (cf. Sec. VI-A). Accounting for this effect in worst-case bounds is possible, but it requires global recalculations whenever new streams enter the network. To prevent signaling storms in decentralized architectures, traffic-independent bounds for burstiness cascading are required. Following our previous work [78], this tutorial utilizes the reservation protocol’s ‘accumulated maximum latency’ field to achieve this – a technique shared by only three other listed publications [5, 75, 76]. Only four publications feature a explicitly distinguished ‘decentral’ latency analysis [5, 76–78]; two of these focus primarily on path finding in delay-constrained networks, while the others restrict their scope to CBS and SP shaping, respectively. Generally, most delay analyses investigate only a single shaper, though an extensive

NC-based performance comparison of SP, ATS, TAS, and CBS is available in [81], albeit without a decentralized focus.

### Lessons learned

Most existing scientific publications focus on GCL scheduling optimization or Network Calculus-based latency analysis. However, current latency analysis literature leaves a significant gap between formal models and practical deployment. Implementing Network Calculus models can be highly challenging when the required information sources remain unclarified. This tutorial complements prior work by strengthening the decentralized dimension of worst-case latency analysis. It explicitly details the mathematical models, demonstrates local information acquisition, provides reference implementations for each model, and compares their admission control efficiency. No other known publication spans these categories.

### III. NOTION AND NOTATION

This section serves as a reference for commonly used terms and symbols throughout this paper. The tutorial is designed to be self-contained, hence this section does not require active reading, but it can be consulted if subsequent equations need clarification.

First, several terms are used synonymously to improve readability. *Frame* and *packet* both refer to a continuous layer-2 Ethernet protocol data unit. Similarly, *delay* and *latency* are used interchangeably. The term *latency jitter* denotes the *maximum possible jitter*, i.e., the total range between maximum and minimum end-to-end latency, rather than standard deviations or differences between consecutive frames. Here, *shaper* refers to a Transmission Selection Algorithm (TSA) from IEEE 802.1Q [161], specifically Strict Priority (SP), Asynchronous Traffic Shaping (ATS), Cyclic Queuing and Forwarding (CQF), or Credit-Based Shaping (CBS). The latter is also known as the Credit-Based Shaping Algorithm (CBSA). The acronym *NC* stands for *Network Calculus*. Most latency models in this tutorial rely on a pre-configured *latency threshold*, also called a *latency configuration* or *latency budget* in literature. It represents the pre-configured upper bound  $\delta_{\mathcal{P}}^{\text{seg}}$  for worst-case latency on an observed link *seg* and priority  $\mathcal{P}$ .

Important distinctions exist between seemingly similar terms. Unlike the threshold  $\delta_{\mathcal{P}}^{\text{seg}}$ , the computed delay  $d_{\mathcal{P}}^{\text{seg}}$  is the active worst-case estimate for already reserved traffic. Admission control enforces  $d_{\mathcal{P}}^{\text{seg}} \leq \delta_{\mathcal{P}}^{\text{seg}}$ ; any reservation exceeding this is rejected. Furthermore, *priority*  $\mathcal{P}$  and *traffic class*  $\mathcal{C}$  are related, but not the same thing. Each stream belongs to a specific priority  $\mathcal{P}$  included in its reservation announcement, and each latency threshold  $\delta_{\mathcal{P}}^{\text{seg}}$  is configured for a specific priority  $\mathcal{P}$ . Both terms operate in the control plane before or during reservation. While IEEE 802.1Q supports up to eight priorities [161], switches may not implement all eight in the data plane. A local mapping table maps priority  $\mathcal{P}$  to a local traffic class  $\mathcal{C}$  on each device. During queuing and forwarding, packets within the same class  $\mathcal{C}_i$  are treated equally important, even if they map to different priorities  $\mathcal{P}_x, \mathcal{P}_y$  in the control plane. Consequently, single-device latency analysis focuses on the local traffic class  $\mathcal{C}$ . Without loss of generality, this paper

TABLE II  
OVERVIEW OF USED SYMBOLS AND ABBREVIATIONS

| Symbol                                       | Ref.       | Description                                                                                                                                   |
|----------------------------------------------|------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| TSN                                          | Sec. I     | Time-Sensitive Networking, an IEEE standardization working group                                                                              |
| RAP                                          | Sec. I     | Resource Allocation Protocol, IEEE 802.1DD                                                                                                    |
| SP                                           | Sec. VI-A  | Strict Priority (no shaping)                                                                                                                  |
| ATS                                          | Sec. VI-D  | Asynchronous Traffic Shaping                                                                                                                  |
| CBS/A                                        | Sec. VI-E  | Credit-Based Shaping (Algorithm)                                                                                                              |
| CQF                                          | Sec. VI-C  | Cyclic Queuing and Forwarding                                                                                                                 |
| TAS                                          | Sec. VI-C  | Time-Aware Shaping                                                                                                                            |
| NC                                           | Sec. VI-B  | Network Calculus, delay calculation framework                                                                                                 |
| <b>Priorities and delay segments</b>         |            |                                                                                                                                               |
| $\mathcal{P}$                                | Sec. III   | Priority 0..7 of a stream (4,5,6,7 for real-time)                                                                                             |
| $\mathcal{C}$                                | Sec. III   | Traffic class of a stream in the data plane of a specific device, based on $\mathcal{P}$ (conversion table)                                   |
| $\mathcal{C}_i^{\text{up}}$                  | Sec. V-B   | Traffic class of stream $i$ in the previous <i>upstream</i> device                                                                            |
| $\hat{\mathcal{C}}$                          | Sec. VI-A  | Highest traffic class of observed device                                                                                                      |
| $\text{prv}$                                 | Fig. 5     | Previous delay segment                                                                                                                        |
| $\text{seg}$                                 | Fig. 5     | Currently observed delay segment                                                                                                              |
| $\text{nxt}$                                 | Fig. 5     | Next delay segment                                                                                                                            |
| $\delta_{\mathcal{P}}^{\text{seg}}$          | Sec. V-A   | Pre-configured delay threshold for priority $\mathcal{P}$ on delay segment <i>seg</i>                                                         |
| $d_{\mathcal{P}}^{\text{seg}}$               | Sec. V-A   | Calculated delay bound for successfully reserved streams for priority $\mathcal{P}$ on delay segment <i>seg</i>                               |
| <b>Traffic Specification (TSPEC)</b>         |            |                                                                                                                                               |
| $\mathcal{L}_i^-$                            | Sec. V-B   | Accumulated minimum latency of stream $i$                                                                                                     |
| $\mathcal{L}_i^+$                            | Sec. V-B   | Accumulated maximum latency of stream $i$                                                                                                     |
| $\ell_i^{\text{min}}$                        | Sec. V-B   | Minimum frame length of stream $i$                                                                                                            |
| $\ell_i^{\text{max}}$                        | Sec. V-B   | Maximum frame length of stream $i$                                                                                                            |
| $\ell^{\text{max}}$                          | Sec. VI-A  | Maximum frame length of all streams on all priorities (typically MTU, e.g., 1542 bytes)                                                       |
| $R^{\text{seg}}$                             | Sec. V-B   | Bandwidth (service rate) of delay segment <i>seg</i>                                                                                          |
| $b_i$                                        | Sec. V-B   | Committed burstiness of stream $i$ (at the talker)                                                                                            |
| $r_i$                                        | Sec. V-B   | Committed data rate of stream $i$ (at the talker)                                                                                             |
| <b>Latency Computation</b>                   |            |                                                                                                                                               |
| $w_i(\Delta t)$                              | Eq. 1      | Traffic volume (workload) generated by stream $i$ during time duration $\Delta t$                                                             |
| $b_i^{\text{eff}}$                           | Eq. 4      | Maximum <i>effective</i> burstiness of stream $i$ at observed device                                                                          |
| $\mathcal{H}_{\mathcal{C}}$                  | Eq. 6      | Set of all streams $i$ with $\mathcal{C}_i > \mathcal{C}$                                                                                     |
| $b_{\mathcal{H}_{\mathcal{C}}}$              | Eq. 18     | Sum of all effective bursts $b_i^{\text{eff}}$ with $i \in \mathcal{H}_{\mathcal{C}}$                                                         |
| $r_{\mathcal{H}_{\mathcal{C}}}$              | Eq. 18     | Sum of all data rates $r_i$ with $i \in \mathcal{H}_{\mathcal{C}}$                                                                            |
| $\mathcal{S}^{\text{seg}}$                   | Alg. 2     | Set of all successfully reserved streams at observed device                                                                                   |
| $\hat{\mathcal{S}}$                          | Eq. 11     | Abbreviation for $\mathcal{S}_{\mathcal{C}}^{\text{seg}}$ : set of all streams that traverse link <i>seg</i> with traffic class $\mathcal{C}$ |
| $\hat{\mathcal{S}}_k$                        | Alg. 4     | Set of streams $\mathcal{S}_{\mathcal{C}}^{\text{prv}_k}$ from the $k$ -th ingress segment $\text{prv}_k$                                     |
| $t_b$                                        | Eq. 10     | Bending position of a single stream arrival curve $\alpha_i$ , after depleting the effective burst $b_i^{\text{eff}}$                         |
| $t_{\hat{b}}$                                | Eq. 11     | Bending position of an aggregate arrival curve $\alpha_{\hat{\mathcal{S}}}$ for all streams of an ingress <i>prv</i>                          |
| $t_{\hat{c}r}$                               | Eq. 30     | 2nd bending position for CBS arrival curves, after depleting positive credits $\text{maxcr}_{\mathcal{C}}^{\text{prv}}$                       |
| $\alpha_i(\Delta t)$                         | Eq. 8      | Arrival curve function for a single stream $i$                                                                                                |
| $\alpha_{\hat{\mathcal{S}}}(\Delta t)$       | Eq. 12     | Aggregate arrival curve of all streams in $\hat{\mathcal{S}}$                                                                                 |
| $\beta_{\mathcal{C}}^{\text{seg}}(\Delta t)$ | Eq. 15     | Service curve for delay segment <i>seg</i> and class $\mathcal{C}$                                                                            |
| $\tau_{\mathcal{C}}^{\text{seg}}$            | Eq. 15     | Initial delay of rate-latency service curve $\beta_{\mathcal{C}}^{\text{seg}}$                                                                |
| $\text{id}_{\mathcal{C}}^{\text{seg}}$       | Eq. 28     | Idle slope of CBS delay segment <i>seg</i> and class $\mathcal{C}$                                                                            |
| $\text{maxcr}_{\mathcal{C}}^{\text{seg}}$    | Eq. 33     | Upper bound for accumulated, positive CBS credits of class $\mathcal{C}$ on delay segment <i>seg</i>                                          |
| <b>Evaluation</b>                            |            |                                                                                                                                               |
| $\mathcal{S}_a^{\mathcal{P}}$                | Sec. VII-A | Set of all <i>accepted</i> streams in priority $\mathcal{P}$                                                                                  |
| $b_a^{\mathcal{P}}$                          | Sec. VII-A | Sum of all accepted streams' bursts in priority $\mathcal{P}$                                                                                 |
| $r_a^{\mathcal{P}}$                          | Sec. VII-A | Sum of all accepted streams' data rates in prio. $\mathcal{P}$                                                                                |
| $\mathcal{L}_i^{\Delta}$                     | Sec. VII-A | Max. latency jitter at receiver, $\mathcal{L}_i^{\Delta} = \mathcal{L}_i^+ - \mathcal{L}_i^-$                                                 |
| $t_{\text{comp}}$                            | Sec. VII-A | Time needed to compute all latency bounds of a scenario                                                                                       |

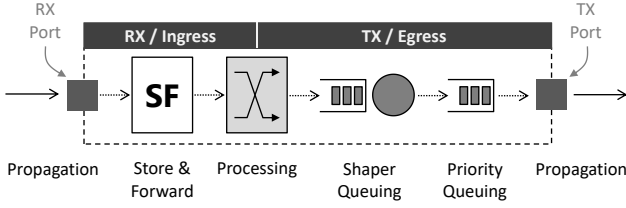


Fig. 4. Switch delay model and different delay types

assumes switches support all eight traffic classes, with the highest four (7, 6, 5, 4) dedicated to real-time traffic.

Table II provides a comprehensive reference list of symbols used in the equations throughout this paper.

#### IV. DELAY MODEL AND DELAY SEGMENTS

To derive worst-case latency upper bounds in switched Ethernet networks, a formal service model covering all relevant latency sources is necessary. For decentralized execution, devices must independently calculate the worst-case delay of adjacent links, which requires well-defined boundaries of responsibility. Accordingly, Section IV-A details the latency model and its delay types, and Section IV-B defines the per-hop delay segment boundaries.

##### A. Types of Delay

Switch latency is traditionally divided into four components [162, Chapter 1.4.1]: transmission (store and forward), propagation, queuing, and processing delay. Figure 4 maps these components to the forwarding process from the perspective of a frame.

*Transmission delay* (store and forward) is the time required to push a frame's bits onto the wire, or read them at the receiver. This paper accounts for it solely at the receiver. It equals the frame size divided by link bandwidth. *Propagation delay* is the time needed to traverse the physical distance between nodes, determined by link length and the medium's propagation speed. *Queuing delay* represents the time a frame waits in an egress queue before transmission, which depends on the service discipline and competing traffic in other queues. Finally, *processing delay* covers internal switch tasks like filtering, metering, table lookups, egress port selection, and MAC/PHY transitions. This component varies heavily based on switch architecture, configuration, and traffic composition.

In Figure 4, queuing delay is split into two phases. *Shaper queueing* encompasses artificial delays introduced by IEEE 802.1Q Transmission Selection Algorithms (TSAs) [161], including Credit-Based, Asynchronous Traffic, and Time-Aware Shaping. Once a frame becomes eligible for transmission under the shaper and any timed-gating mechanisms, it enters *priority queueing*. Here, the queue of the highest traffic class that is currently not empty is serviced at line rate. Isolating these two phases is necessary for the definition of delay segments for decentralized scenarios.

Note that this model serves as a behavioral abstraction for latency analysis, without implying certain physical switch implementations. The two logical queues can be implemented in the same physical buffer, the model only separates the queuing process by its cause.

##### B. Delay Segments

Decentralized latency models require clear boundaries of responsibility along a path. The following approach has been proposed as a contribution [163, 164] to the IEEE P802.1DD RAP standard [7].

Intuitively, segments would be split at physical device boundaries, e.g., from ingress to ingress. However, insights from the analysis of Asynchronous Traffic Shaping [165] show that placing measurement points inside devices can facilitate latency computation. Specifically, it analyzed the delay from shaper to shaper: from the moment a frame becomes eligible at an upstream shaper until it becomes eligible at the downstream shaper. This ensures well-defined traffic workloads at each segment boundary. For instance, traffic entering a segment from a token-bucket shaper strictly conforms to those shaping parameters before experiencing interference in the priority queue. The resulting segments are illustrated in Figure 5. During reservation, the downstream switch computes the worst-case latency bound for its incoming segment.

Beyond aligning with existing ATS analyses, this approach offers broader benefits. Shapers inherently restore formal traffic properties, such as enforcing token-bucket volume bounds (ATS) or reducing jitter to combat burstiness cascading (CQF). Although subsequent priority queuing reintroduces jitter, the downstream shaper can compensate for it. Defining segments post-shaping allows analyses to leverage these formal traffic properties directly, turning the shaper's output into a well-defined input for the next segment. This uniform definition also simplifies shaper transitions in heterogeneous networks.

##### C. Lessons learned

The assumption that the borders of latency analysis must match with the physical borders of the devices can be a limitation for tight hop-by-hop latency bounds. This paper separates the queuing process into two stages: shaping and priority queuing. The delay segment of a single hop is defined from one shaper's end to following shaper's end. This shift allows to analyze the delay of that segment in a decentralized architecture, while profiting from the formal, predictable behavior of the shaper in the analysis.

#### V. INFORMATION FLOW

When streams require deterministic service guarantees, this is typically achieved by reserving network resources for them. These resources include physical capacities (bandwidth, buffer space) and abstract metrics like worst-case latency. Every new reservation increases worst-case latency, effectively consuming the available 'latency budget'. Consequently, dynamic admission control rejects reservations that would push worst-case latency beyond a pre-defined threshold, much like it manages bandwidth limits.

To calculate the exact latency consumed by each stream, the models from Section VI are used. These models rely on stream and neighbor device information to compute upper bounds for individual delay segments. This section details this information and its propagation, presenting a simplified signaling model based on the Resource Allocation Protocol (RAP) [7] draft, and is meant to further support its development.

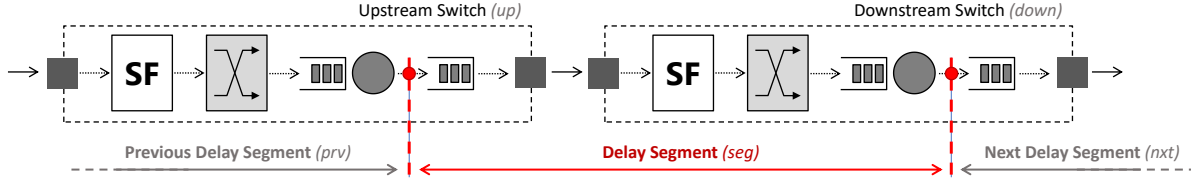


Fig. 5. Delay segments: borders of responsibility for the decentralized per-hop latency computation (components are introduced in Fig. 4)

TABLE III  
INFORMATION EXCHANGED BY SWITCHES BEFORE AND DURING STREAM RESERVATION

| Shared Information                                                                                    | Purpose                                                                                          | Origin             | Notes                                      |
|-------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|--------------------|--------------------------------------------|
| <b>A Priori / Before Reservation</b>                                                                  |                                                                                                  |                    |                                            |
| Latency threshold $\delta_{\mathcal{P}}^{seg}$ per delay segment $seg$ and per priority $\mathcal{P}$ | Added to each reservation's AccMaxLatency, and used as an upper bound for resource consumption   | Pre-configured     | Not actually shared, just required locally |
| Used shaper per traffic class                                                                         | Estimation of burstiness of incoming traffic                                                     | Neighbor's config. | —                                          |
| CBS idle slope (shaper parameter)                                                                     | Improved CBS latency bound                                                                       | Neighbor's config. | Only applicable to CBS                     |
| CQF cycle time and GCL offsets (shaper parameter)                                                     | Verify matching configuration                                                                    | Neighbor's config. | Only applicable to CQF                     |
| <b>During Reservation</b>                                                                             |                                                                                                  |                    |                                            |
| Stream ID and Ethernet header fields                                                                  | Identification, priority, matching of incoming packets                                           | Per stream         | —                                          |
| Traffic Specification (TSpec)                                                                         | Specification of traffic amount                                                                  | Per stream         | —                                          |
| Accumulated latency (min $\mathcal{L}^-$ , max $\mathcal{L}^+$ )                                      | Estimation of burstiness increase from latency jitter                                            | Per stream         | Increased at every hop                     |
| Upstream traffic class $\mathcal{C}^{up}$                                                             | Computation of worst-case queuing delay in the upstream switch, based on its internal priorities | Per stream         | —                                          |
| Ingress port number of upstream device                                                                | Advanced Strict Priority latency bound (Sec. VI-B)                                               | Per stream         | Only for improved SP                       |
| Previous delay segment link speed $R^{prv}$                                                           | Advanced Strict Priority latency bound (Sec. VI-B)                                               | Per link           | Only for improved SP                       |

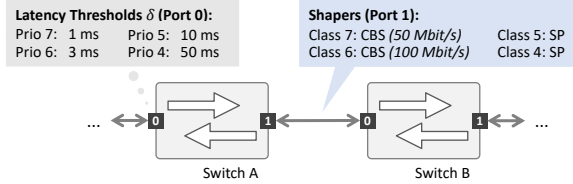


Fig. 6. Exemplary a priori information flow: Switch A shares its shaper configuration of port 1 with Switch B, its direct neighbor

### A. A Priori Information

Before signaling information about streams, certain device-specific parameters must be exchanged first. These are summarized in the first part of Table III, with an exchange example shown in Figure 6, and detailed below.

**Latency Threshold:** Most latency algorithms in Section VI rely on pre-configured latency thresholds  $\delta_{\mathcal{P}}^{seg}$  per delay segment  $seg$  and priority  $\mathcal{P}$ . For each segment, the downstream switch handles latency computations. During reservation, it verifies that the active worst-case latency  $d_{\mathcal{P}}^{seg}$  does not exceed  $\delta_{\mathcal{P}}^{seg}$ . This threshold is added to each stream's accumulated maximum latency  $\mathcal{L}_i^+$ , which represents a traffic-independent upper bound for the end-to-end latency of that stream which remains valid even after future reservations. This accumulated latency is later used to evaluate future stream reservations and compute latency bounds further down the stream's path (cf. Equation 4 in Section VI-A). Notably,  $d_{\mathcal{P}}^{seg}$  is calculated dynamically for admission control to measure resource consumption, whereas the configured threshold  $\delta_{\mathcal{P}}^{seg}$  serves as the absolute resource limit and is advertised to neighbors as a guarantee. Modifying  $\delta_{\mathcal{P}}^{seg}$  post-hoc could invalidate existing reservations.

In this work, thresholds are defined per priority  $\mathcal{P}$  rather

than per traffic class  $\mathcal{C}$  because a delay segment spans two devices that may not support identical internal traffic classes. Although the RAP draft [7, Sec. 6.4.2.2] introduces a shared neighbor-to-neighbor class called *RAClass*, we omit it here for simplicity and use the Ethernet VLAN priority  $\mathcal{P}$  instead.

**Upstream Shaper:** Computing tight bounds requires the downstream switch to know the shaper configurations of the upstream device. In Figure 6, Switch A informs Switch B that port 1 uses the Credit-Based Shaper for classes 7 and 6, and Strict Priority for classes 5 and 4 (classes 3–0 handle Best-Effort traffic and are omitted here). Certain shapers require additional parameter sharing: the Credit-Based Shaper requires the idle slope for tight bounds (cf. Section VI-E and [5]), while Cyclic Queuing and Forwarding requires matching cycle timings (cf. Section VI-C).

Note that shaper configurations only need to be shared between direct neighbors to let the downstream switch estimate incoming burstiness. While the RAP draft supports this via *RA class templates* [7, Annex D] in the *RA Class Descriptor* [7, Sec. 6.4.2.2], it does not yet distribute specific shaper parameters.

### B. Per-Stream Information

Calculating worst-case queuing latency requires bounding the queue traffic volume. End devices must share stream characteristics to estimate traffic volume along the path. This information, as summarized in Table III and illustrated in Figure 7, falls into four categories: stream identification, Traffic Specification, accumulated latency, and upstream traffic class.

**Stream Identification:** Streams must be uniquely identified during reservation and subsequent data forwarding.

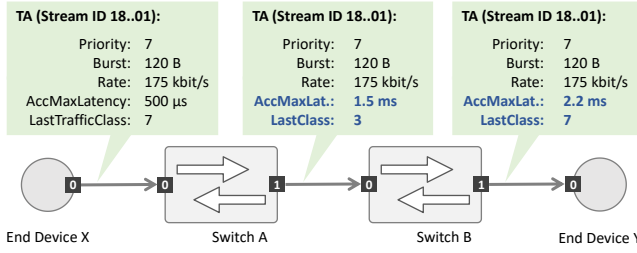


Fig. 7. Exemplary information flow during stream reservation: Talker Announcement (TA) from End Device X to End Device Y (not all info shown)

RAP uses a unique stream ID combining the sender's MAC address and a unique number. Because data plane Ethernet headers lack a dedicated stream ID field, active packets are matched using previously signaled header fields: destination MAC address, VLAN ID, and VLAN priority code point. Figure 7 simplifies this ID to 18..01. The assigned priority  $\mathcal{P}$  identifies the relevant delay threshold  $\delta_{\mathcal{P}}^{\text{seg}}$  during reservation and maps to a local traffic class  $\mathcal{C}$  for queue selection.

**Traffic Specification:** To estimate queue occupancy, the Traffic Specification must bound the traffic volume over any time interval. While the legacy SRP standard [166] used frame counts within a class measurement interval, this fits poorly with modern, token-bucket-based latency algorithms. Translating between these models is possible, but it introduces an overhead and reduces network efficiency. Thus, RAP defaults to a token-bucket model, which is also adopted here. Each stream  $i$  specifies a burst size  $b_i$ , data rate  $r_i$ , and maximum/minimum frame lengths ( $\ell_i^{\text{max}}, \ell_i^{\text{min}}$ ). For any duration  $\Delta t$ , the traffic volume  $w_i$  of stream  $i$  is bounded by [167]:

$$w_i(\Delta t) \leq b_i + r_i \cdot \Delta t. \quad (1)$$

Here,  $r_i$  is the long-term rate and  $b_i$  is the maximum short-term burst deviation from this rate. In Figure 7, only  $b_i$  and  $r_i$  are shown. The 'burst' does not necessarily need to arrive immediately in one piece. In graphical terms, it is the maximum deviation of the cumulative traffic from the linear function  $y(\Delta t) = r_i \cdot \Delta t$ . Note that  $b_i \geq \ell_i^{\text{max}}$ , and all parameters ( $b_i, r_i, \ell_i^{\text{max}}$ ) must include layer-1 overhead – typically 20 B for preamble, CRC, and inter-frame gap.

**Accumulated Latency:** The above Traffic Specification only captures the behavior at the sender, but a stream's traffic characteristics can change as it encounters network delays. To track this, reservation protocols carry the *accumulated maximum latency*  $\mathcal{L}_i^+$  and *accumulated minimum latency*  $\mathcal{L}_i^-$  to compute the effective burstiness inside the network (cf. Equation 1). Each hop updates these fields before propagation:

$$\mathcal{L}_i^+(\text{Downstream}) = \mathcal{L}_i^+(\text{Upstream}) + \delta_{\mathcal{P}_i}^{\text{seg}}, \quad (2)$$

$$\mathcal{L}_i^-(\text{Downstream}) = \mathcal{L}_i^-(\text{Upstream}) + \ell_i^{\text{min}} / R^{\text{seg}}. \quad (3)$$

Here,  $R^{\text{seg}}$  is the physical link speed of the segment,  $\ell_i^{\text{min}}$  is the minimum frame size of stream  $i$ , and  $\delta_{\mathcal{P}_i}^{\text{seg}}$  is the local delay threshold for priority  $\mathcal{P}$ . The difference ( $\mathcal{L}_i^+ - \mathcal{L}_i^-$ ) represents the maximum latency jitter accumulated along the path.

**Previous Delay Segment Link Speed:** For the improved Strict Priority latency bound in Section VI-B, additionally, the ingress link speed  $R^{\text{prv}}$  of the link  $\text{prv}$  to the previous upstream

switch is required. This value is carried in the reservation data and updated at each hop, using  $R^{\text{prv}} = \infty$  for the first segment. Note that the basic SP model does not require this parameter (cf. Eq. 6).

**Upstream Traffic Class:** Because the downstream switch calculates the upstream priority queuing delay, it must know each stream's upstream traffic class assignment  $\mathcal{C}_i^{\text{up}}$ . Switches do not always support all eight standard classes [161], and priority-to-class mapping can vary across devices. Thus, reservations should explicitly carry the upstream traffic class. Each switch overrides this field with its local value for the next hop. In Figure 7, Switch A (with limited class support) maps the stream to class 3, while Switch B maps it to class 7. Switch B must use Switch A's actual class mapping rather than assuming its own configuration during calculations.

### C. Lessons learned

Legacy reservation protocols like SRP only convey limited information, and although *accumulated maximum latency* already exists in SRP, it was not strictly enforced via admission control. There was a selection of example latency formulas in the standards [168, Sec. 6.5], but they lacked formal proof and were shown to be inaccurate [5, 169]. This shows that signaling must be based on a solid mathematical foundation.

Decentralized worst-case delay computation requires signaling of local state parameters. Switches must know their direct neighbors' shaping configurations. For individual streams, signaling must convey the original Traffic Specification and the accumulated maximum latency on its path to calculate the localized, effective traffic volume. Because the real, tight accumulated latency would fluctuate frequently when reservations change, a static, pre-configured maximum threshold is propagated instead. Under shaper-to-shaper segments, the downstream switch calculates the upstream queuing delay, which means the upstream device must share its local priority-to-class mappings. Tighter delay bounds can be achieved by also leveraging the previous device's ingress port speed, though this is not yet supported by standard signaling protocols.

## VI. LATENCY BOUNDS

With the signaling information from Section V established, all prerequisites for decentralized worst-case latency calculations are met. This section details the mathematical models and provides pseudo-code implementations to evaluate the worst-case latency of reserved traffic. Note that these models focus on the *transmission* and *queuing delays* described in Section IV-A. For *propagation* and *processing delays*, devices are assumed to possess independent upper bounds derived from their specific hardware configurations.

First, Section VI-A addresses the basic scenario of Strict Priority transmission selection without shaping, deriving a simple formula for practical implementation. Section VI-B then refines this bound using the Network Calculus framework [167, 170–172] by accounting for physical link speed serialization. Specifically, frames arriving from the same upstream port cannot be received simultaneously. Next, Section VI-C adapts the basic Strict Priority formula for Cyclic

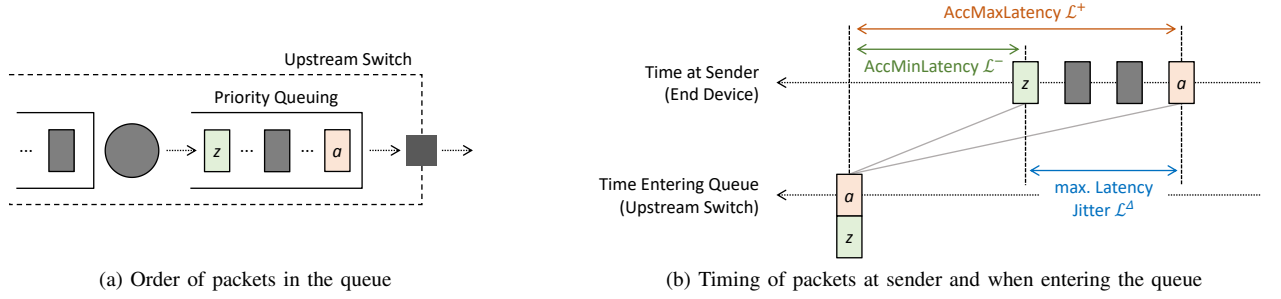


Fig. 8. Timings and delays of packets that can be in the priority queue at the same time

Queuing and Forwarding, while Section VI-D outlines the latency analysis for Asynchronous Traffic Shaping proposed by Specht and Samii [165]. Finally, Section VI-E adapts the refined Strict Priority model to accommodate the unique characteristics of the Credit-Based Shaper.

#### A. Strict Priority – Simple Bound

Under Strict Priority (SP) transmission selection, the shaper queuing delay from Figure 4 is zero, meaning downstream devices introduce no artificial delays. Consequently, the delay of that segment is primarily caused by the priority queue of the upstream switch within the same segment. To compute this delay, the downstream switch needs to know the maximum *effective burstiness* of all reserved streams when they enter that priority queue.

While end devices declare their original burstiness in the Traffic Specification, the effective burstiness can increase further downstream due to accumulated latency jitter. In simple terms, frames with lower latency travel ‘faster’ and catch up to frames with higher latency, which travel ‘slower’. This can cause packets that were originally sent separately to cluster in downstream queues. The maximum data accumulation from this effect can be bounded using the maximum latency jitter, given by the difference between the accumulated maximum and minimum latency fields.

Our prior work [78] used the same accumulated latency fields under a more constrained Traffic Specification, based on packet inter-arrival times. In contrast, the effective burstiness bound formulated in this work is more generic and directly applicable to standard token-bucket-shaped traffic.

**Maximum Effective Burstiness:** First, this paragraph analyzes the maximum traffic volume that can arrive ‘instantaneously’ in the upstream switch’s priority queue, i.e., at a rate exceeding the Traffic Specification rate  $r_i$ . This burstiness bound assumes that end-device transmissions conform to their Traffic Specification, and that the worst-case latency of each delay segment remains within its pre-configured threshold, which is enforced via admission control during stream reservation.

For any stream  $i$  with Traffic Specification  $w_i$  (Eq. 1), the accumulated maximum latency  $\mathcal{L}_i^+$ , and accumulated minimum latency  $\mathcal{L}_i^-$  from the previous delay segment, the maximum effective burst  $b_i^{\text{eff}}$  is bounded by:

$$b_i^{\text{eff}} \leq w_i(\mathcal{L}_i^+ - \mathcal{L}_i^-) = b_i + r_i \cdot (\mathcal{L}_i^+ - \mathcal{L}_i^-). \quad (4)$$

Here,  $b_i^{\text{eff}}$  represents the amount of bytes that can theoretically be enqueued simultaneously in the priority queue, after shaping, as illustrated in Figure 8. This priority queue is depicted in Figure 8a, where packets are ordered from earliest arrival  $a$  to latest  $z$ . Furthermore, Figure 8b illustrates the packet timings from both the sender’s and the priority queue’s perspectives. Note that the time advances from right to left to retain the order from the queue. Any packet sent after  $z$  cannot contribute to  $b_i^{\text{eff}}$  because its minimum latency would fall below the  $\text{AccMinLatency } \mathcal{L}_i^-$ . Similarly, packets sent before  $a$  cannot contribute to  $b_i^{\text{eff}}$  because their latency would exceed the  $\text{AccMaxLatency } \mathcal{L}_i^+$ . Consequently, the relevant time interval  $\Delta t$  for  $w_i(\Delta t)$  in Equation 1 is defined by the difference between these two accumulated latency fields. A comparable property can be formally derived using the Network Calculus framework, cf. [5, Eq. 20].

**Latency Bound for the Highest Traffic Class:** From the perspective of the highest traffic class  $\hat{C}$  of the upstream switch, the following observations can be made.

(1) During any busy period of class  $\hat{C}$ , only a single lower-class transmission can occur, which must begin before the first higher-class packet arrives in the queue. Thereafter, the higher-class queue is completely emptied before any lower-class packet is transmitted again.

(2) If the maximum lower-class packet length is  $\ell^{\text{max}}$  and the segment link speed is  $R^{\text{seg}}$ , this lower-class transmission introduces a latency of  $\ell^{\text{max}} / R^{\text{seg}}$ .

(3) During this lower-class transmission, at most  $b_i^{\text{eff}}$  bytes arrive from any stream  $i \in \hat{C}$ , so their worst-case latency becomes  $(\sum_{i \in \hat{C}} b_i^{\text{eff}} + \ell^{\text{max}}) / R^{\text{seg}}$  until now.

(4) While servicing these initial bursts, additional higher-class packets may arrive with rate  $r_i$ . These subsequent packets do not experience higher latency because admission control ensures that the link speed exceeds the aggregate rate:  $\sum_{i \in \hat{C}} r_i < R^{\text{seg}}$ . This allows the backlog to clear faster than it accumulates. Furthermore, these packets do not delay preceding packets as they are enqueued behind them.

In summary, for the highest traffic class  $\hat{C}$ , the worst case latency  $d_{\hat{C}}^{\text{seg}}$  is bounded by:

$$d_{\hat{C}}^{\text{seg}} \leq \frac{\sum_{i \in \hat{C}} b_i^{\text{eff}} + \ell^{\text{max}}}{R^{\text{seg}}}. \quad (5)$$

**Latency Bound for Lower Traffic Classes:** Consider any observed priority  $\mathcal{P}$ , whose delay bound is desired. Using information from prior signaling, the downstream switch maps

it to the upstream traffic class  $\mathcal{C} = \mathcal{C}_{\mathcal{P}}^{up}$ . For any traffic class  $\mathcal{C}$  below the highest class, the analysis is extended as follows.

(5) Ignoring all higher-priority traffic classes  $\mathcal{H}_{\mathcal{C}}$  yields the same latency bound:  $(\sum_{i \in \mathcal{C}} b_i^{eff} + \ell^{max})/R^{seg}$ .

(6) Packets from bursts in higher classes  $\mathcal{H}_{\mathcal{C}}$  are prioritized over class  $\mathcal{C}$ , adding an extra delay of  $\sum_{i \in \mathcal{H}_{\mathcal{C}}} b_i^{eff}/R^{seg}$ .

(7) Unlike same- or lower-priority packets, new packets from higher classes  $\mathcal{H}_{\mathcal{C}}$  take precedence over  $\mathcal{C}$  even if they arrive later. This increases the worst-case delay of class  $\mathcal{C}$  based on the higher streams' data rates  $r_i$ . During any interval  $\Delta t$ , the workload increases by  $\Delta t \cdot r_i$ . Due to admission control, the relevant time interval  $\Delta t$  will never exceed the delay threshold  $\delta_{\mathcal{P}}^{seg}$ , so the maximum interference from any higher stream  $i$  is bounded by  $\delta_{\mathcal{P}}^{seg} \cdot r_i$ . Note that this bound holds when the actual worst-case delay  $d_{\mathcal{P}}^{seg}$  remains below  $\delta_{\mathcal{P}}^{seg}$ , a condition verified for every reservation during admission control.

In summary, for any priority  $\mathcal{P}$  and its corresponding upstream traffic class  $\mathcal{C} = \mathcal{C}_{\mathcal{P}}^{up}$ , the worst case latency  $d_{\mathcal{P}}^{seg}$  is bounded by:

$$d_{\mathcal{P}}^{seg} \leq \frac{\sum_{i \in \mathcal{H}_{\mathcal{C}}} (b_i^{eff} + r_i \cdot \delta_{\mathcal{P}}^{seg}) + \sum_{i \in \mathcal{C}} b_i^{eff} + \ell^{max}}{R^{seg}} \quad (6)$$

Here,  $\mathcal{C}$  is the set of streams in the upstream traffic class of the observed priority  $\mathcal{P}$ ,  $\mathcal{H}_{\mathcal{C}}$  represents higher-priority streams,  $b_i^{eff}$  is the effective burstiness of stream  $i$  (Eq. 4),  $\delta_{\mathcal{P}}^{seg}$  is the pre-configured latency threshold of the delay segment,  $\ell^{max}$  is the maximum length of any lower-priority packet, and  $R^{seg}$  is the egress link speed.

*Pseudo Code Implementation:* Implementing Equation 6 is relatively straightforward, provided the downstream and upstream traffic classes and their associated delay segments are not confused. For the lower-priority packet length, a full Maximum Transmission Unit (MTU) must typically be assumed to account for unknown Best-Effort traffic. Here,  $\ell^{max} = 1542$  Bytes is used, comprising a layer-2 MTU of 1522 Bytes and 20 Bytes of layer-1 overhead (preamble, CRC, and inter-frame gap).

---

**Algorithm 1:** max\_interf

---

**Input:** Stream  $i$ , Priority  $\mathcal{P}$ , Delay Segment  $seg$

```

1 if  $\mathcal{C}_i^{up} == \mathcal{C}_{\mathcal{P}}^{up}$  then // same class
2   | return  $w_i(\mathcal{L}_i^+ - \mathcal{L}_i^-)$ 
3 else if  $\mathcal{C}_i^{up} > \mathcal{C}_{\mathcal{P}}^{up}$  then // higher class
4   | return  $w_i(\mathcal{L}_i^+ - \mathcal{L}_i^- + \delta_{\mathcal{P}}^{seg})$ 
5 else // lower class
6   | return 0
```

---

First, Algorithm 1 computes the individual contribution of every stream to the latency of the observed priority  $\mathcal{P}$  across three cases. If stream  $i$  and priority  $\mathcal{P}$  share the same class, the algorithm returns the effective burstiness  $b_i^{eff}$  from Equation 4. If the stream's class is higher, it incorporates the extra interference  $\delta_{\mathcal{P}}^{seg} \cdot r_i$  by adding  $\delta_{\mathcal{P}}^{seg}$  to the interval  $\Delta t$  in  $w_i(\Delta t)$ . If the class is lower, it returns zero, as lower-class interference is handled separately in Algorithm 2. Note that all traffic classes  $\mathcal{C}$  (including the target priority's class  $\mathcal{C}_{\mathcal{P}}^{up}$ )

refer to the upstream switch. Additionally, the accumulated latencies  $\mathcal{L}_i^+$  and  $\mathcal{L}_i^-$  denote the values prior to being updated via Equations 2 and 3.

---

**Algorithm 2:** wc\_delay\_sp\_simple

---

**Input:** Priority  $\mathcal{P}$ , Delay Segment  $seg$ , Streams  $\mathcal{S}^{seg}$ , Additional Stream  $s$

```

1 interf ←  $\ell^{max}$ 
2 for Stream  $i \in \mathcal{S}^{seg} \cup \{s\}$  do
3   | interf ← interf + max_interf( $i, \mathcal{P}$ )
4 return interf /  $R^{seg}$ 
```

---

Next, Algorithm 2 evaluates the numerator of Equation 6 by summing  $\max\_interf(i, \mathcal{P})$  from Algorithm 1 for all streams in the segment. Lower class interference  $\ell^{max}$  is only added once. This total interference is then divided by link speed  $R^{seg}$  to complete Equation 6. This process evaluates the existing reservations  $\mathcal{S}^{seg}$  in addition to a candidate stream  $s$  awaiting admission control.

This simple implementation highlights a key benefit of the basic Strict Priority bound: it supports incremental computation. Because subsequent stream reservations do not alter  $\ell^{max}$  or  $\max\_interf(i, \mathcal{P})$  for existing streams, implementations can cache the latency bounds of  $\mathcal{S}^{seg}$  and perform only a single addition for a new candidate stream  $s$ . This can significantly reduce overhead on low-power switch CPUs, but this optimization is omitted in this paper for simplicity. In the following section, a tighter bound is suggested at the expense of a higher computation time.

### B. Strict Priority – Tighter Bound

The latency bound in Equation 6 simplifies the computation by assuming that all bursts from all streams arrive simultaneously. In practice, without shaping, this is only possible if every burst originates from a different ingress port. The opposite extreme is a system with one ingress port and one egress port operating at identical link speeds. Here, the maximum latency remains  $\ell^{max}/R^{seg}$  regardless of the stream count or burstiness, because the egress queue drains at the same rate that incoming data arrives. This section presents a generalized latency bound that accounts for this data serialization at ingress ports across all intermediate configurations.

This is achieved by analyzing the maximum arriving bits and the minimum serviced bits over time as separate cumulative functions, where the worst-case latency is the maximum difference between them. This approach is commonly formalized using the Network Calculus framework. To keep this paper self-contained, the relevant Network Calculus theorems are only stated briefly. For comprehensive proofs and derivations, see [167, 170–172].

In short, the core idea is to model the priority queue's arrival and service processes using piecewise linear curves. The maximum horizontal distance between these curves yields the worst-case delay. The closer these curves represent the real system, the tighter the delay bound becomes. The arrival curves are constructed from the Traffic Specifications

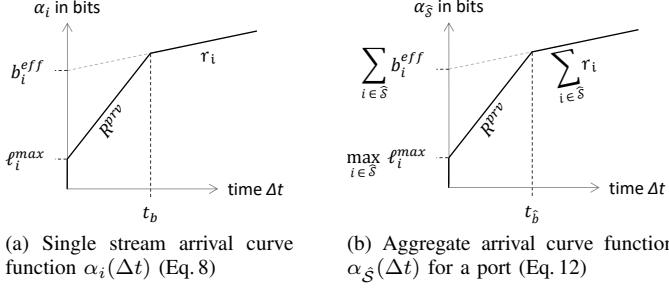


Fig. 9. Piecewise linear arrival curves for token bucket traffic models

(Section V-B) while accounting for stream distribution across physical ports. The service curves are modeled as rate-latency curves, where the initial latency captures the waiting time imposed by higher-priority traffic.

**Arrival Curves:** Let  $\mathcal{A}$  be the arrival process of one or more streams at a port, where  $\mathcal{A}(t_1, t_2) = \mathcal{A}(t_2) - \mathcal{A}(t_1)$  denotes the data volume arriving in the interval  $(t_1, t_2]$ . This process is upper-bounded by an arrival curve  $\alpha(\Delta t) = \alpha(t_2 - t_1)$  if, for all  $0 \leq t_1 \leq t_2$ :

$$\mathcal{A}(t_1, t_2) \leq \alpha(t_2 - t_1). \quad (7)$$

For a token-bucket-shaped stream  $i$  with burst  $b_i$  and rate  $r_i$ , the baseline arrival curve at the source is  $\alpha_i(\Delta t) = w_i(\Delta t) = b_i + r_i \cdot \Delta t$  for  $\Delta t \geq 0$ . However, at any downstream switch, the effective burstiness increases to  $b_i^{\text{eff}}$  (Eq. 4). Furthermore, the instantaneous arrival of data is bounded by the maximum length of a single frame  $\ell_i^{\text{max}}$ . The remainder of a burst will arrive at the link speed  $R^{\text{prv}}$  of the upstream segment  $\text{prv}$ . Thus, the arrival curve for stream  $i$  in segment  $\text{seg}$  is modeled as:

$$\alpha_i(\Delta t) = \begin{cases} \ell_i^{\text{max}} + R^{\text{prv}} \cdot \Delta t & \text{if } \Delta t \leq t_b, \\ b_i^{\text{eff}} + r_i \cdot \Delta t & \text{else.} \end{cases} \quad (8)$$

Here,  $t_b$  is the time when the arrival rate transitions from line rate  $R^{\text{prv}}$  to the stream rate  $r_i$  (Fig. 9a). It is determined by the intersection of two simpler curves. Both  $\alpha_{i,1}(\Delta t) = b_i^{\text{eff}} + r_i \cdot \Delta t$  and  $\alpha_{i,2}(\Delta t) = \ell_i^{\text{max}} + R^{\text{prv}} \cdot \Delta t$  are valid upper bounds individually, where  $\alpha_{i,1}$  is the limit of produced data at the source, and  $\alpha_{i,2}$  is the limit of transmission speed on the link. Their minimum is also an upper bound, and the intersection point  $t_b$  of these functions is:

$$b_i^{\text{eff}} + r_i \cdot t_b = \ell_i^{\text{max}} + R^{\text{prv}} \cdot t_b \quad (9)$$

$$\Leftrightarrow t_b = \frac{b_i^{\text{eff}} - \ell_i^{\text{max}}}{R^{\text{prv}} - r_i}. \quad (10)$$

For the first delay segment where the upstream device is the traffic source,  $R^{\text{prv}} = \infty$  can be assumed, simplifying the model to  $\alpha_{i,1}(\Delta t) = b_i^{\text{eff}} + r_i \cdot \Delta t$ .

For the full arrival curve of the entire delay segment  $\text{seg}$ , the superposition property of Network Calculus could be used, which would basically sum up all individual stream arrival curves. However, this would overestimate the immediate arrivals as  $\sum_{i \in \mathcal{C}} \ell_i^{\text{max}}$ , i.e., the sum of all streams' frame lengths. In reality, simultaneous frame arrivals are physically limited by the number of ingress ports. To obtain a tighter bound, one aggregate arrival curve per ingress port is constructed

(Fig. 9b), and the full arrival curve will be the sum of those curves instead of the individual stream curves.

Let  $\hat{\mathcal{S}} = \mathcal{S}_C^{\text{prv}}$  represent all streams of class  $\mathcal{C}$  arriving from the previous segment  $\text{prv}$ . Because  $\text{prv}$  represents a single port, immediate data arrival is limited to a single maximum frame length,  $\max_{i \in \hat{\mathcal{S}}} \ell_i^{\text{max}}$ . Thereafter, data can arrive at line rate  $R^{\text{prv}}$  for a duration of  $t_b$ , after which the upstream queue is empty and the arrival rate drops to the aggregate source rate  $\sum_{i \in \hat{\mathcal{S}}} r_i$ . Similarly to before, the aggregate arrival curve  $\alpha_{\hat{\mathcal{S}}}$  is the minimum of two curves. First,  $\alpha_{\hat{\mathcal{S}},1}(\Delta t) = \sum_{i \in \hat{\mathcal{S}}} (b_i^{\text{eff}} + r_i \cdot \Delta t)$  is the upper bound of cumulative data produced by all traffic sources. Second,  $\alpha_{\hat{\mathcal{S}},2}(\Delta t) = \max_{i \in \hat{\mathcal{S}}} \ell_i^{\text{max}} + R^{\text{prv}} \cdot \Delta t$  is the limit enforced by the ingress link speed. These two curves intersect at:

$$t_b = \frac{\sum_{i \in \hat{\mathcal{S}}} b_i^{\text{eff}} - \max_{i \in \hat{\mathcal{S}}} \ell_i^{\text{max}}}{R^{\text{prv}} - \sum_{i \in \hat{\mathcal{S}}} r_i}. \quad (11)$$

The resulting piecewise linear curve (Fig. 9b) is defined by:

$$\alpha_{\hat{\mathcal{S}}}(\Delta t) = \begin{cases} \max_{i \in \hat{\mathcal{S}}} \ell_i^{\text{max}} + R^{\text{prv}} \cdot \Delta t & \text{if } \Delta t \leq t_b, \\ \sum_{i \in \hat{\mathcal{S}}} b_i^{\text{eff}} + \sum_{i \in \hat{\mathcal{S}}} r_i \cdot \Delta t & \text{else.} \end{cases} \quad (12)$$

The total arrival curve  $\alpha_C^{\text{seg}}$  for segment  $\text{seg}$  and class  $\mathcal{C}$  is the sum of the aggregate arrival curves across all  $n$  active ingress ports  $\text{prv}_k$ :

$$\alpha_C^{\text{seg}}(\Delta t) = \sum_{1 \leq k \leq n} \alpha_{\hat{\mathcal{S}}_k}(\Delta t). \quad (13)$$

Here, the sum iterates over all  $n$  ingress ports  $\text{prv}_k$  (cf. Fig. 5) using stream sets  $\hat{\mathcal{S}}_k = \mathcal{S}_C^{\text{prv}_k}$ . For strict priority systems, the upstream switch's priority queue delay is analyzed. This requires all streams from  $\bigcup_{\text{prv}} \mathcal{S}_C^{\text{prv}}$  across all previous segments  $\text{prv}$ . These streams do not necessarily traverse the observed segment  $\text{seg}$  downstream, they can continue on different delay segments, but they share the same upstream priority queue.

**Service Curves:** Now, after the arrival process is determined, a formal model for worst-case service is required, given by a *minimum* cumulative service curve. As before, it is based on the signaling information from Section V. The general Network Calculus methodologies are detailed in [167, 171, 172], and briefly presented in the following.

Let  $\mathcal{A}$  be the arrival process of one or more streams at a service unit, and  $\mathcal{A}(t_1, t_2) = \mathcal{A}(t_2) - \mathcal{A}(t_1)$  be the amount of data that arrives from that process in the time interval  $(t_1, t_2]$ . Further, let  $\mathcal{A}^*$  be the departure process of that service unit, and  $\mathcal{A}^*(t_1, t_2) = \mathcal{A}^*(t_2) - \mathcal{A}^*(t_1)$  be the amount of data that departs from that process in the time interval  $(t_1, t_2]$ . The service unit has a service curve  $\beta(\Delta t) = \beta(t_2 - t_1)$  if, for all  $0 \leq t_1 \leq t_2$ , the following equation holds:

Let  $\mathcal{A}$  and  $\mathcal{A}^*$  be the cumulative arrival and departure processes at a service unit, and let  $(t_1, t_2]$  be an arbitrary time interval. During that time,  $\mathcal{A}(t_2) - \mathcal{A}(t_1)$  bytes arrived at the service unit, and  $\mathcal{A}^*(t_2) - \mathcal{A}^*(t_1)$  bytes were serviced and departed. The unit offers a service curve  $\beta(\Delta t) = \beta(t_2 - t_1)$  if, for all  $0 \leq t_1 \leq t_2$ :

$$\mathcal{A}^*(t_2) \geq \inf_{t_1 \leq t_2} \{\mathcal{A}(t_1) + \beta(t_2 - t_1)\}. \quad (14)$$

Less formally, Equation 14 guarantees that at least  $\beta(\Delta t)$  bits are serviced during any backlogged interval  $\Delta t$ .

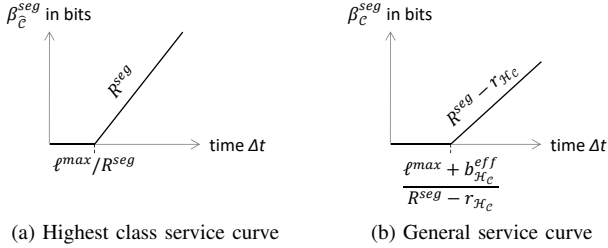


Fig. 10. Service curves for Strict Priority and Token Bucket traffic model

A typical, linear service model is the rate-latency service curve  $\beta_C^{seg}$  with a constant rate  $R_C^{seg}$  and an initial latency  $\tau_C^{seg}$ :

$$\beta_C^{seg}(\Delta t) = R_C^{seg} \cdot (\Delta t - \tau_C^{seg}). \quad (15)$$

To capture the physical service of link  $seg$  to class  $C$ , the two parameters  $R_C^{seg}$  and  $\tau_C^{seg}$  must be defined. These parameters are influenced by higher-priority traffic arrivals, which reduce lower-class service guarantees. Note that service curves typically cannot become negative. Without loss of generality, terms like  $\max(0, \dots)$  are omitted here for simplicity, with no impact on their applicability.

*Service Curve for the Highest Traffic Class:* For the highest class  $\hat{C}$ , the service rate  $R_{\hat{C}}^{seg}$  is the full link speed  $R^{seg}$ , as its queue  $Q_{\hat{C}}$  drains without interruption once it is selected for transmission. However, if a lower-class packet *just* started its own transmission before  $Q_{\hat{C}}$  contained any packets, it can delay the start of  $\hat{C}$  up to the length of a maximum transmission unit  $\ell^{max}$ . The highest-class service curve is therefore given by:

$$\beta_{\hat{C}}^{seg}(\Delta t) = R^{seg} \cdot \left( \Delta t - \frac{\ell^{max}}{R^{seg}} \right) = R^{seg} \cdot \Delta t - \ell^{max}. \quad (16)$$

This simple linear function is illustrated in Figure 10a.

*Service Curve for Lower Traffic Classes:* For a general traffic class  $C$ , let  $\mathcal{H}_C$  be the set of higher-class streams. Packets from class  $C$  must wait until any ongoing transmission finishes and all queues from higher classes  $\mathcal{H}_C$  are empty. Based on Equation 4, higher-class queues hold at most  $b_{\mathcal{H}_C}^{eff} = \sum_{i \in \mathcal{H}_C} b_i^{eff}$  bytes. In the worst case, queue  $C$  waits for these  $\ell^{max} + b_{\mathcal{H}_C}^{eff}$  bytes to transmit at rate  $R^{seg}$ . Meanwhile, new higher-class data arrives at rate  $r_{\mathcal{H}_C} = \sum_{i \in \mathcal{H}_C} r_i$ , which must also be served first. Thus, higher-class queues empty at an effective rate of  $R^{seg} - r_{\mathcal{H}_C}$ . Applying this rate to both the service rate  $R_C^{seg}$  and the delay term  $\tau_C^{seg}$  yields:

$$\beta_C^{seg}(\Delta t) = R_C^{seg} \cdot (\Delta t - \tau_C^{seg}) \quad (17)$$

$$= (R^{seg} - r_{\mathcal{H}_C}) \cdot \left( \Delta t - \frac{\ell^{max} + b_{\mathcal{H}_C}^{eff}}{R^{seg} - r_{\mathcal{H}_C}} \right). \quad (18)$$

Equation 18 is further illustrated in Figure 10b.

*Latency Bound:* The Network Calculus delay theorem states that the worst-case delay for class  $C$  on segment  $seg$  is bounded by the maximum horizontal deviation  $h$  between  $\alpha_C^{seg}$  and  $\beta_C^{seg}$ :

$$d_P^{seg} \leq h(\alpha_C^{seg}, \beta_C^{seg}) \quad (19)$$

$$= \sup_{s \geq 0} \{ \inf \{ \tau \geq 0 : \alpha_C^{seg}(s) \leq \beta_C^{seg}(s + \tau) \} \}. \quad (20)$$

Practically, this bound can be found by comparing slopes, cf. Figure 11. The piecewise arrival curve  $\alpha_C^{seg}$  has a continuously

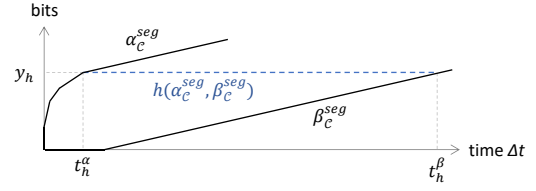


Fig. 11. Max. horizontal distance between arrival and service curve

decreasing slope. Consider sweeping an imaginary horizontal line from the bottom to the top of Figure 11. The distance between  $\alpha_C^{seg}$  and  $\beta_C^{seg}$  increases as long as the slope of  $\alpha_C^{seg}$  is higher than the service rate  $R_C^{seg}$ . Assuming the segment is not overprovisioned, the arrival slope eventually drops below or meets  $R_C^{seg}$ , after which the horizontal distance no longer increases. Thus, the maximum delay occurs where the arrival slope equals  $R_C^{seg}$ . This peak is marked by  $t_h^\alpha$  on the x-axis and  $y_h = \alpha_C^{seg}(t_h^\alpha)$  on the y-axis. The corresponding service curve point is  $t_h^\beta$ , where  $\beta_C^{seg}(t_h^\beta) = y_h$  holds. The maximum delay is then given by their horizontal difference  $d_P^{seg} \leq t_h^\beta - t_h^\alpha$ .

In this work, a closed form description of the maximum horizontal distance is omitted. Instead, the pseudo code implementation obtains the delay bound numerically by looping through the individual pieces of  $\alpha_C^{seg}$  as explained above, until the slope is smaller or equal to  $R_C^{seg}$ .

*Pseudo Code Implementation:* First, data structures for the arrival and service curves are required. The arrival curve  $\alpha_{\hat{S}}(\Delta t)$  represents streams from a single ingress segment  $prv$  and a single class  $C$ . Based on Equation 12 and Figure 9b, it is implemented as a tuple:  $ac_{\hat{S}} = (burst, slope1, slope2, bendpos)$ . Here,  $burst = \max_{i \in \hat{S}} \ell_i^{max}$ ,  $slope1 = R^{prv}$ , and  $slope2 = \sum_{i \in \hat{S}} r_i$ . The two slopes transition at  $bendpos = t_b$ , as defined in Equation 11. The function value is computed as follows:

$$\alpha_{\hat{S}}(\Delta t) = \begin{cases} burst + slope1 \cdot \Delta t & \text{if } \Delta t \leq t_b, \\ \alpha_{\hat{S}}(t_b) + slope2 \cdot (\Delta t - t_b) & \text{else.} \end{cases} \quad (21)$$

Note that the *burst* in Equation 21 is based on immediately arriving traffic from the perspective of the ingress port. It is given by the frame lengths  $\ell_i^{max}$ , and not by the burstiness  $b_i$  of streams, which is incorporated in  $t_b$  (Eq. 11). The slope of  $\alpha_{\hat{S}}$  at any point  $\Delta t$  is a simple case statement:

$$slope_{\hat{S}}(\Delta t) = \begin{cases} slope1 & \text{if } \Delta t < t_b, \\ slope2 & \text{else.} \end{cases} \quad (22)$$

Similarly, the service curve  $\beta_C^{seg}(\Delta t)$  from Equation 18 is represented by  $sc_C^{seg} = (slope, offset)$ , where  $slope = R_C^{seg}$  and  $offset = \tau_C^{seg}$ . Its value at time  $\Delta t$  is  $sc_C^{seg}(\Delta t) = \max(0, slope \cdot (\Delta t - offset))$ . More importantly, its inverse is given by:

$$findt_C^{seg}(y) = y/slope + offset. \quad (23)$$

The full arrival curve, as defined in Equation 13, is represented by the list  $AC_C^{seg} = [ac_{\hat{S},1}, ac_{\hat{S},2}, \dots, ac_{\hat{S},n}]$ , where the  $k$ -th tuple represents streams from the  $k$ -th ingress port:  $\mathcal{S}_C^{prv,k}$ . The downstream switch must know this stream-to-port mapping and the corresponding link speeds  $R^{prv} = ac_{\hat{S},k}.slope1$ , which must be provided during stream reservation, cf. Table III.

Given the list of tuples  $AC_C^{seg}$ , the function  $\text{slope}_C^{seg}(\Delta t) = \sum_{1 \leq k \leq n} \text{slope}_{\hat{s}_k}(\Delta t)$  returns the aggregate slope at point  $\Delta t$ . Its inverse,  $\text{find\_slope}_C^{seg}(s)$ , returns the earliest time  $\Delta t$  where the aggregate slope is at most  $s$ . Algorithm 3 implements this by looping through a sorted list of bending positions until the aggregate slope falls below  $s$ .

---

**Algorithm 3: find\_slope**


---

**Input:** Class  $\mathcal{C}$ , Delay Segment  $seg$ , Slope  $s$ , List of arrival curve tuples  $AC$

```

1 // slope tuples: (slope, bendpos)
2  $SL \leftarrow []$  // empty list of tuples
3 //  $SL = [(s1, b1), (s2, b2), \dots]$ 
4 // initialize first tuple at  $t=0$ 
5  $SL.append((\sum_{ac \in AC} ac.slope1, 0))$ 
6 for  $ac \in AC$  do
7    $bendpos \leftarrow ac.bendpos$ 
8    $slope \leftarrow \text{slope}_C^{seg}(bp)$  //  $\sum \text{slopes}(bp)$ 
9    $SL.append((slope, bendpos))$ 
10 sort( $SL, key = bendpos$ )
11  $i \leftarrow 0$ 
12 while  $SL[i].slope > s$  do // search for  $s$ 
13    $i \leftarrow i + 1$ 
14   if  $i \geq |SL|$  then return  $-1$  // error
15 return  $SL[i].bendpos$ 
```

---

Ultimately, using arrival curves  $AC_C^{seg}$  and service curve  $sc_C^{seg}$ , the two points  $t_h^\alpha$  and  $t_h^\beta$  via  $\text{findt}$  must be found, cf. Figure 11. Here,  $t_h^\alpha$  is given when the arrival rate falls below the service rate and can be obtained via  $\text{find\_slope}$ . Then,  $t_h^\beta$  can be found via  $\text{findt}_C^{seg}()$ . Finally, the delay bound is  $d_P^{seg} \leq t_h^\beta - t_h^\alpha$ , as implemented in Algorithm 4.

---

**Algorithm 4: wc\_delay\_sp**


---

**Input:** Priority  $\mathcal{P}$ , Delay Segment  $seg$ , List of streams per ingress port  $LS_C^{seg} = [\hat{S}_1, \dots, \hat{S}_n]$

```

1  $\mathcal{C} \leftarrow \mathcal{C}_P^{up}$  // upstream traffic class
2  $AC \leftarrow []$  // empty list
3 for  $\hat{S}_k \in LS$  do
4    $ac \leftarrow (0, 0, 0, 0)$  // prepare tuple
5    $ac.burst \leftarrow \max_{i \in \hat{S}_k} \ell_i^{max}$ 
6    $ac.slope1 \leftarrow R_{\hat{S}_k.prv}$  //  $R^{prv}$ 
7    $ac.slope2 \leftarrow \sum_{i \in \hat{S}_k} r_i$ 
8    $ac.bendpos \leftarrow t_b$  // Eq. 11
9    $AC.append(ac)$ 
10  $sc \leftarrow (0, 0)$  // slope, offset
11  $sc.slope \leftarrow R^{seg} - \sum_{i \in \mathcal{H}_C} r_i$ 
12  $sc.offset \leftarrow \frac{\ell^{max} + \sum_{i \in \mathcal{H}_C} b_i^{eff}}{R^{seg} - \sum_{i \in \mathcal{H}_C} r_i}$ 
13  $t_h^\alpha \leftarrow \text{find\_slope}_C^{seg}(sc.slope, AC)$ 
14  $d_P^{seg} \leftarrow \text{findt}_C^{seg}(\alpha_C^{seg}(t_h^\alpha)) - t_h^\alpha$ 
15 return  $d_P^{seg}$ 
```

---

In summary, the delay bound in Algorithm 4 is tighter than the simple bound in Algorithm 2, but it is more complex,

slower to compute, and requires more information from the reservation process, cf. Table III. The comparison is picked up in Section VII in more detail.

### C. Cyclic Queuing and Forwarding

Cyclic Queuing and Forwarding (CQF) [161, Annex T] is the first shaper addressed in this work. This section outlines the modifications introduced by CQF, and then addresses how the latency model of SP can be adapted to that.

Standardization-wise, CQF is not a standalone shaper but a specific configuration of Time-Aware Shaping (TAS) and Per-Stream Filtering and Policing (PSFP), as illustrated in Figure 12. For detailed derivations of these TSN mechanisms, readers are referred to the standard [161] and comprehensive surveys [8]. Briefly, *cyclic queuing* periodically alternates traffic of the same priority between multiple queues. Incoming streams are directed to one queue by PSFP, while another queue is selected by TAS for transmission. Both gating mechanisms require globally synchronized time, and identical configurations are deployed on every network device. An exemplary configuration is shown in Figure 13 and analyzed below.

The primary objective is to ensure that every hop-to-hop transmission takes exactly  $T$  time units, even under light traffic load. To achieve this, the gating mechanisms of PSFP and TAS are operated out-of-phase. When PSFP selects an Internal Priority Value (IPV) of 7, the corresponding transmission gate is closed by TAS, while the gate for 6 is opened. This state is illustrated in Figure 12. After a fixed slot duration of  $T$ , PSFP assigns an IPV of 6 to incoming packets of the same stream, while TAS closes gate 6 and opens gate 7 (depicted in Figure 13). Consequently, as soon as TAS opens a gate on Switch A, PSFP on Switch B classifies these frames to its closed queue. Because this closed gate is opened exactly  $T$  time units after the upstream gate on Switch A, each hop between shapers takes exactly  $T$  time units. Thus, the latency threshold  $\delta$  (cf. Sec. V-A) must be configured as  $\delta \leq T - d_{misc}$ , where  $d_{misc}$  accounts for synchronization inaccuracies. If the admission control mechanism ensures that the worst-case delay  $d$  remains below  $\delta$ , the queue is guaranteed to be empty at the end of window  $T$ . Hence, a downstream shaper considers a frame eligible for transmission exactly  $T$  time units after the upstream shaper did. Leveraging this property is a key benefit of the shaper-to-shaper delay segments presented in Section IV-B.

*Latency Analysis:* Adapting the simple SP latency model from Section VI-A to CQF involves three key modifications.

(i) The number of usable traffic classes is halved, because each external class requires two internal queues. Without loss of generality, the evaluation in Section VII assumes eight available traffic classes, where classes 4 to 7 are allocated to reserved traffic. Under CQF, classes are consolidated to the higher priorities, i.e.,  $\mathcal{P}_{CQF}(7) = 7, \mathcal{P}_{CQF}(6) = 7, \mathcal{P}_{CQF}(5) = 5, \mathcal{P}_{CQF}(4) = 5$ . This consolidation increases traffic interference in high-priority classes, which raises their worst-case delay. For a more sophisticated evaluation, priority mappings and delay bounds could be optimized specifically for CQF.

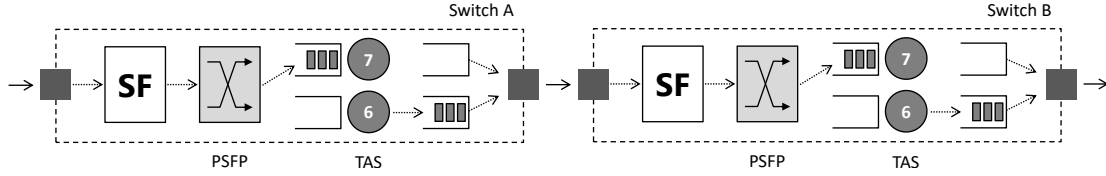


Fig. 12. Snapshot of a Cyclic Queuing and Forwarding configuration in this work's delay model (first time slot from Figure 13)

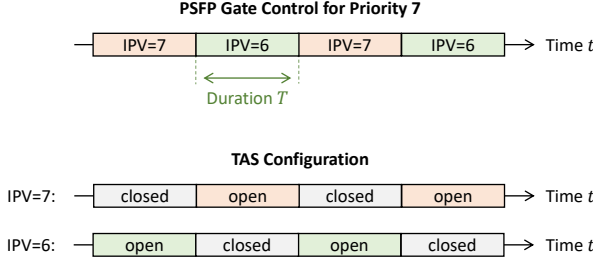


Fig. 13. Example configuration of PSFP and TAS gates for one external priority in CQF with Internal Priority Values (IPVs) 6 and 7

(ii) In the shaper-to-shaper delay model, the minimum latency becomes equal to the maximum latency. Equation 3 for  $\mathcal{L}^-$  is thus updated to match  $\mathcal{L}^+$ :

$$\mathcal{L}_{i,CQF}^+(Downstream) = \mathcal{L}_{i,CQF}^+(Upstream) + T_{P_i}, \quad (24)$$

$$\mathcal{L}_{i,CQF}^-(Downstream) = \mathcal{L}_{i,CQF}^-(Upstream) + T_{P_i}. \quad (25)$$

Consequently, in Equation 4, the effective burst  $b_i^{eff}$  is nearly identical to the original burst  $b_i$ , depending solely on the difference between  $\mathcal{L}^-$  and  $\mathcal{L}^+$  at the first hop.

(iii) The SP model assumes that backlog is cleared faster than new traffic accumulates. This assumption does not hold for CQF, where the gating mechanism effectively stops transmissions until queues alternate. Consequently, extra traffic can accumulate in the queue of priority  $\mathcal{P}$  for up to  $\delta_{\mathcal{P}}^{seg}$  time units. This additional volume is calculated similarly to the higher-priority case as  $\delta_{\mathcal{P}}^{seg} \cdot r_i$ .

---

#### Algorithm 5: max\_interf\_cqf

---

**Input:** Stream  $i$ , Priority  $\mathcal{P}_{orig}$ , Delay Segment  $seg$

```

1  $\mathcal{P} \leftarrow \mathcal{P}_{CQF}(\mathcal{P}_{orig})$ 
2 if  $C_i^{up} \geq C_{\mathcal{P}}^{up}$  then           // higher class
3   | return  $w_i(\mathcal{L}_i^+ - \mathcal{L}_i^- + \delta_{\mathcal{P}}^{seg})$ 
4 else                               // lower class
5   | return 0

```

---

Accounting for these adjustments, Algorithm 5 details the adaptation of the original max\_interf (Algorithm 1) to the CQF priority mapping. The remainder of the SP latency analysis from Section VI-A remains unchanged. Detailed formulas and pseudo-codes are omitted. Note that the simple SP formula must be used for CQF, as gating negates the serialization gains modeled by the improved formula. The enhancements from Section VI-B are inapplicable here, as the final TAS stage in the segment would require a very different service curve. Furthermore, periodic streams can be penalized by the token bucket traffic model. For instance, a periodic

stream transmitting 120 bytes packets every 2 ms would be mapped to a burst of 120 bytes and a rate of 480 kbit/s. If the CQF window is configured to be 2 ms as well, in theory, there should only be one packet per CQF window. However, Algorithm 5 yields at least 240 bytes of worst-case interference. This is a valid assumption under the token bucket model, which permits a second packet of, e.g., 110 bytes after only 1.9 ms, before the queues alternate. For such scenarios, incorporating specialized traffic models for periodic streams would be highly beneficial.

#### D. Asynchronous Traffic Shaping

The shift of delay segments described in Section IV-B was first proposed in the latency analysis of the Asynchronous Traffic Shaper (ATS) [173] in [165]. This section briefly outlines the changes introduced by ATS, and presents its latency analysis results. Further details are available in [165].

ATS enforces the token-bucket constraint from Equation 1 using per-stream shaping. Within each switch, each stream  $i$  is assigned a virtual token bucket with capacity  $b_i$  and token generation rate  $r_i$ . A frame  $f_i$  of size  $\ell_{f_i}$  can only be transmitted if  $\ell_{f_i}$  tokens can be subtracted from the bucket without going negative. Every shaper in this work is assumed to be configured to re-enforce the end device's original burst and rate constraints. Individual shaping combined with per-class queuing can cause head-of-line blocking delays, therefore ATS mitigates this by applying three Queue Allocation Rules to separate traffic into virtual queues before the shaper. Specifically, frames are isolated into separate virtual queues if they have different ingress port, egress port, or traffic class on either the upstream or downstream switch.

*Latency Analysis:* Under these constraints, the worst-case latency with ATS is bounded by:

$$d_{\mathcal{P}}^{seg} \leq \frac{\sum_{i \in \mathcal{H}_C} b_i + \sum_{i \in \mathcal{C}} b_i - \ell_{\mathcal{P}}^{min} + \ell_{\mathcal{P}}^{max}}{R^{seg} - \sum_{i \in \mathcal{H}_C} r_i} + \frac{\ell_{\mathcal{P}}^{min}}{R^{seg}}. \quad (26)$$

This formulation simplifies the original equation in [165, Eq. 21]. Unlike the simple SP model in Equation 6, the ATS delay bound uses the original burstiness from the end devices' Traffic Specification. Furthermore, ATS avoids the need for a pre-configured latency threshold  $\delta_{\mathcal{P}}^{seg}$ , though its queuing rules require more complex hardware implementations. Finally, the serialization-based tighter bounds from Section VI-B are not applicable to ATS, as a limited amount of head-of-line blocking can still occur within the virtual queues.

*Pseudo Code:* Because this latency model is independent of accumulated latencies ( $\mathcal{L}^-$ ,  $\mathcal{L}^+$ ) and latency thresholds ( $\delta_{\mathcal{P}}^{seg}$ ), its implementation is simpler than Algorithms 1 and 2. The corresponding logic is detailed in Algorithm 6. The

**Algorithm 6:** wc\_delay\_atc

---

**Input:** Priority  $\mathcal{P}$ , Streams  $\mathcal{S}^{seg}$ , Additional Stream  $s$

```

1  $interf \leftarrow \ell_P^{max}$ 
2  $r_{\mathcal{H}_C} \leftarrow 0$ 
3  $\ell_P^{min} \leftarrow \ell_P^{max}$ 
4 for Stream  $i \in \mathcal{S}^{seg} \cup \{s\}$  do
5   if  $C_i^{up} == C_P^{up}$  then
6      $interf \leftarrow interf + b_i$ 
7      $\ell_P^{min} \leftarrow \min(\ell_P^{min}, \ell_i^{min})$ 
8   else if  $C_i^{up} > C_P^{up}$  then
9      $interf \leftarrow interf + b_i$ 
10     $r_{\mathcal{H}_C} \leftarrow r_{\mathcal{H}_C} + r_i$ 
11 return  $(interf - \ell_P^{min}) / (R^{seg} - r_{\mathcal{H}_C}) + \ell_P^{min} / R^{seg}$ 

```

---

implementation uses a single loop over all reserved streams, including the new reservation candidate, to update the total interfering burst ( $interf$ ), the sum of higher-priority rates ( $r_{\mathcal{H}_C}$ ), and the minimum frame lengths of the target priority ( $\ell_P^{min}$ ). These computed metrics are then used to evaluate and return the latency bound from Equation 26.

**E. Credit-Based Shaping**

Predating the Asynchronous Traffic Shaper, the Credit-Based Shaper (CBS) [161, Sec.8.6.8.2] was originally designed to limit egress traffic to provide solid mathematical foundations for worst-case latency analysis. However, the latency model proposed in the 802.1BA profile [168, Sec. 6.5] was proven inaccurate [5, 169]. Because CBS is a class-based shaper, multiple streams are shaped as an aggregate by a single instance. This aggregation introduces latency jitter despite shaping, which leads to an increased effective burstiness as described in Section VI-A.

For each port and traffic class, CBS manages a credit value and a configured rate called the *idle\_slope*. Frame transmission is only permitted when the credit value of that class is non-negative. This credit value increases over time with the *idle\_slope*. During active transmission of a class  $\mathcal{C}$  frame, the credit decreases at rate  $R^{seg}$ , yielding an effective decrease rate of  $R^{seg} - \text{idle\_slope}$ . Although credits can become positive, the value resets to zero if a new frame arrives at an empty queue, preventing long-term credit accumulation. Consequently, positive credits only build up while the queue is non-empty. This behavior resembles the token-bucket model in ATS, with two exceptions: (i) shaping applies to the entire traffic class rather than per-stream, and (ii) the credit value is not upper bounded directly by the shaper parameters, but indirectly by the time that the queue can be non-empty and idle at the same time, due to higher-priority traffic.

**Conformance with Delay Segment Definition:** The CBS is a unique case in this work. The delay segment definition in Section IV-B requires strict separation between shaping and priority transmission selection. This separation is straightforward for ATS and CQF, and it benefits their analysis. However, it is challenging for CBS because its credit reset behavior depends on whether other frames of the same class remain

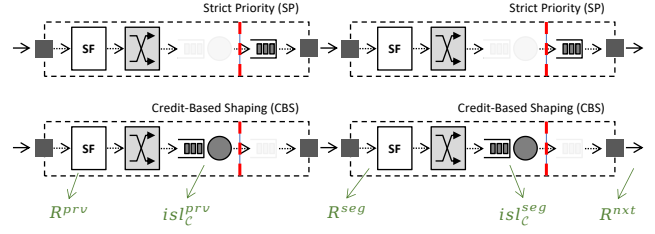


Fig. 14. Comparison of effective delay segments in SP and CBS

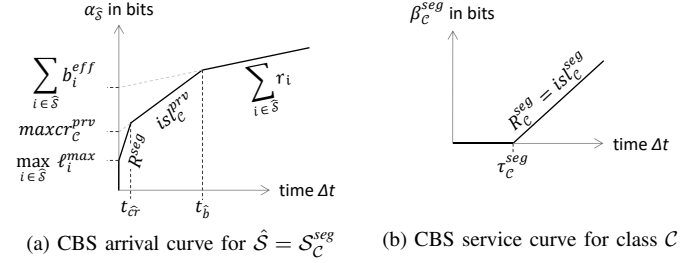


Fig. 15. Adjusted arrival and service curves for CBS

in the queue. Thus, the shaper state is directly influenced by priority transmission selection.

As the shaper state changes continuously during transmission, the *shaping* step from Figure 4 is only completed after a frame is fully transmitted. The *priority queuing* step is empty, similar to the empty *shaping* step in the SP mechanism. This relationship is shown in Figure 14, along with the link speeds  $R^{\{prv, seg, nxt\}}$  and *idle\_slopes*  $isl_C^{\{prv, seg, nxt\}}$ . Under CBS, the downstream switch calculates its own priority queuing delay within the latency model, similar to the behavior of most delay models in literature. Co-existence of CBS with other shapers therefore requires careful handling of transitions between these distinct delay models. This integration challenge is discussed in Section VIII. Here, it is assumed that the upstream switch also employs CBS and has already considered its own queuing delay accordingly.

**Latency Analysis:** The latency analysis of CBS adapts the improved Network Calculus formula for SP from Section VI-B with three main modifications. (i) The delay segment incorporates the queuing delay of the downstream switch rather than the upstream switch. As a consequence, the transmission queue of a single egress port is shared by several delay segments *seg* from different ingress ports. (ii) The piecewise arrival curve of SP (Fig. 9) is adapted to include a segment corresponding to the CBS *idle\_slope*  $isl_C^{prv}$ , which can reduce calculated delays due to restricted traffic arrival. (iii) The rate-latency service curve of SP (Fig. 10) is adapted to the reduced rate of the *idle\_slope*  $isl_C^{seg}$ , which typically increases calculated delays as the service rate decreases.

The resulting CBS curves are summarized in Figure 15 and detailed below.

For the **arrival curve**, the model from Figure 9b is valid but can be refined using CBS bandwidth limits, as shown in Figure 15a. Class  $\mathcal{C}$  transmits at line rate  $R^{seg}$  only while its credit is positive. Thereafter, even if the accumulated burst  $\sum_{i \in \hat{\mathcal{S}}} b_i^{eff}$  is not depleted, the transmission rate is constrained by the upstream device's idle slope  $isl_C^{seg}$ . If this idle slope exceeds the aggregate stream rate  $\sum_{i \in \hat{\mathcal{S}}} r_i$ , the rate drops to

that aggregate rate at the moment  $t_{\hat{b}}$ , just like in the SP model.

The arrival curve is modeled as the infimum of three upper bounds:

$$\alpha_{\hat{S},1}(\Delta t) = \max_{i \in \hat{S}} \ell_i^{\max} + R^{\text{seg}} \cdot \Delta t, \quad (27)$$

$$\alpha_{\hat{S},2}(\Delta t) = \max_{i \in \hat{S}} \text{maxcr}_C^{\text{prv}} + \text{isl}_C^{\text{prv}} \cdot \Delta t, \quad (28)$$

$$\alpha_{\hat{S},3}(\Delta t) = \sum_{i \in \hat{S}} b_i^{\text{eff}} + \sum_{i \in \hat{S}} r_i \cdot \Delta t. \quad (29)$$

These curves intersect at the points  $t_{\hat{c}r}$  and  $t_{\hat{b}}$ , as indicated in Figure 15a. The intersection points are given by:

$$t_{\hat{c}r} = \max \left( 0, \frac{\max_{i \in \hat{S}} \text{maxcr}_C^{\text{prv}} - \max_{i \in \hat{S}} \ell_i^{\max}}{R^{\text{seg}} - \text{isl}_C^{\text{prv}}} \right), \quad (30)$$

$$t_{\hat{b}} = \max \left( t_{\hat{c}r}, \frac{\sum_{i \in \hat{S}} b_i^{\text{eff}} - \max_{i \in \hat{S}} \text{maxcr}_C^{\text{prv}}}{\text{isl}_C^{\text{prv}} - \sum_{i \in \hat{S}} r_i} \right). \quad (31)$$

Here, the term  $\text{maxcr}_C^{\text{seg}}$  represents the maximum positive credit for class  $C$  on segment  $\text{seg}$ . Its computation is revisited below, especially in Equation 33. Note that, if  $\text{maxcr}_C^{\text{seg}} < \max_{i \in \hat{S}} \ell_i^{\max}$  holds, then  $t_{\hat{c}r} = 0$  is assumed, but  $\alpha_{\hat{S}}(\Delta t)$  is always at least equal to  $\max_{i \in \hat{S}} \ell_i^{\max}$  for every non-negative  $t$ . This results in the following curve  $\alpha_{\hat{S}}$  for the streams of a single ingress port  $\hat{S} = \mathcal{S}_C^{\text{prv}}$ :

$$\alpha_{\hat{S}}(\Delta t) = \begin{cases} \max_{i \in \hat{S}} \ell_i^{\max} + R^{\text{seg}} \cdot \Delta t & \text{if } 0 \leq \Delta t \leq t_{\hat{c}r}, \\ \max_{i \in \hat{S}} \text{maxcr}_C^{\text{prv}} + \text{isl}_C^{\text{prv}} \cdot \Delta t & \text{if } t_{\hat{c}r} < \Delta t \leq t_{\hat{b}}, \\ \sum_{i \in \hat{S}} b_i^{\text{eff}} + \sum_{i \in \hat{S}} r_i \cdot \Delta t & \text{if } t_{\hat{b}} < \Delta t. \end{cases} \quad (32)$$

As before, the full arrival curve  $\alpha_C^{\text{seg}}$  is the sum of individual ingress arrival curves  $\alpha_{\hat{S}}$ , cf. Equation 13.

The **service curve** follows the same rate-latency model. As shown in Figure 15b, the service rate  $R_C^{\text{seg}}$  matches the *idle\_slope*  $\text{isl}_C^{\text{seg}}$  of the CBS. The shaper only allows to exceed this rate temporarily, when positive credit has accumulated. In this model, these positive credit phases are omitted, as these high-rate phases can be interrupted by various events. As before, the offset of the service curve can be obtained in two steps.

First, the offset  $\tau_C^{\text{seg}}$  for the highest class  $\hat{C}$  is bounded by one MTU-sized lower-priority packet, identical to the SP analysis:  $\tau_C^{\text{seg}} = \ell_C^{\max} / R^{\text{seg}}$ . For lower classes  $C$ , the offset depends on the maximum duration during which higher classes  $\mathcal{H}_C$  can continuously transmit. The adjacent higher class  $C+1$  accumulates credit for at most  $\tau_{C+1}^{\text{seg}}$  time, reaching up to  $\tau_{C+1}^{\text{seg}} \cdot \text{isl}_{C+1}^{\text{seg}}$  credits. These credits are consumed during transmissions at rate  $R^{\text{seg}} - \text{isl}_{C+1}^{\text{seg}}$ , but they can be interrupted by additional higher class packets at rate  $r_{\mathcal{H}_{C+1}} = \sum_{i \in \mathcal{H}_{C+1}} r_i$ . At zero credit, one additional frame up to size  $\ell_{C+1}^{\max}$  may complete transmission. Together, this results in the following offset  $\tau_C^{\text{seg}}$ :

$$\text{maxcr}_C^{\text{seg}} = \tau_C^{\text{seg}} \cdot \text{isl}_C^{\text{seg}} \quad (33)$$

$$\tau_C^{\text{seg}} = \tau_{C+1}^{\text{seg}} + \frac{\text{maxcr}_{C+1}^{\text{seg}}}{R^{\text{seg}} - \text{isl}_{C+1}^{\text{seg}} - r_{\mathcal{H}_{C+1}}} + \frac{\ell_{C+1}^{\max}}{R^{\text{seg}}}. \quad (34)$$

*Pseudo Code:* The CBS model is implemented based on the improved SP pseudo-code. Arrival curves  $\alpha_{\hat{S}}$  for different ingress ports are represented by the tuple:  $ac_{\hat{S}} =$

$(\text{burst}, \text{slope1}, \text{slope2}, \text{slope3}, \text{bendpos1}, \text{bendpos2})$ . The service curve  $\beta_C^{\text{seg}}$  uses the same representation as before:  $sc_C^{\text{seg}} = (\text{slope}, \text{offset})$ . Algorithm 7 maps this logic to pseudo-code. As input, it takes a list of streams which belong to class  $C = \mathcal{C}_P^{\text{down}}$ , grouped by ingress port at the downstream switch. If two streams were mapped to different traffic classes by upstream, they must be in separate groups  $\hat{S}_x, \hat{S}_y$ , too. Although they cannot arrive simultaneously from the same port, they have different ingress idle slopes, which cannot easily be combined.

---

**Algorithm 7:** wc\_delay\_cbs

---

**Input:** Priority  $\mathcal{P}$ , Delay Segment  $\text{seg}$ , List of streams per ingress port  $LS_C^{\text{seg}} = [\hat{S}_1, \dots, \hat{S}_n]$

```

1  $\mathcal{C} \leftarrow \mathcal{C}_P^{\text{down}}$  // downstream traffic class
2  $AC \leftarrow []$  // empty list
3 for  $\hat{S}_k \in LS$  do
4    $\mathcal{C}_k \leftarrow \mathcal{C}_P^{\text{up}}$ 
5    $ac \leftarrow (0, 0, 0, 0, 0, 0)$  // prepare tuple
6    $ac.\text{burst} \leftarrow \sum_{i \in \hat{S}_k} b_i^{\text{eff}}$ 
7    $ac.\text{slope1} \leftarrow R^{\text{seg}}$ 
8    $ac.\text{slope2} \leftarrow \text{isl}_{\mathcal{C}_k}^{\text{prv}}$  // idle slope
9    $ac.\text{slope3} \leftarrow \sum_{i \in \hat{S}_k} r_i$ 
10   $ac.\text{bendpos1} \leftarrow t_{\hat{c}r}$  // Eq. 30
11   $ac.\text{bendpos2} \leftarrow t_{\hat{b}}$  // Eq. 31
12   $AC.\text{append}(ac)$ 
13  $c \leftarrow \hat{C}$  // highest class
14  $\tau_c \leftarrow \ell_C^{\max} / R^{\text{seg}}$  // initial offset
15 while  $c > \mathcal{C}$  do // recursive function
16    $c \leftarrow c - 1$ 
17    $\tau_c \leftarrow \tau_{c+1}^{\text{seg}} + \frac{\tau_{c+1}^{\text{seg}} \cdot \text{isl}_{c+1}^{\text{seg}}}{R^{\text{seg}} - \text{isl}_{c+1}^{\text{seg}} - r_{\mathcal{H}_{c+1}}} + \frac{\ell_{c+1}^{\max}}{R^{\text{seg}}}$ 
18  $sc \leftarrow (0, 0)$  // slope, offset
19  $sc.\text{slope} \leftarrow \text{isl}_C^{\text{seg}}$ 
20  $sc.\text{offset} \leftarrow \tau_c$ 
21  $t_h^{\alpha} \leftarrow \text{find\_slope\_cbs}_C^{\text{seg}}(sc.\text{slope}, AC)$ 
22  $d_P^{\text{seg}} \leftarrow \text{findt}_C^{\text{seg}}(\alpha_C^{\text{seg}}(t_h^{\alpha})) - t_h^{\alpha}$ 
23 return  $d_P^{\text{seg}}$ 
```

---

Algorithm 7 is similar to the previous Algorithm 4, but it features a larger arrival curve setup. Additionally, the service curve calculation (lines 13–20) is now recursive because  $\tau_c$  depends on  $\tau_{c+1}$ . The final step uses the identical inverse function  $\text{findt}_C^{\text{seg}}(y) = y / \text{slope} + \text{offset}$ , while  $\text{find\_slope\_cbs}_C^{\text{seg}}$  extends Algorithm 3 by processing two bending positions per arrival curve, so the contents of the for-loop (lines 7–9) must be done for both of them.

### F. Lessons learned

In simplified terms, worst-case delay computation for SP, CQF, ATS and CBS follows the same pattern: using Traffic Specifications and accumulated latencies to get an upper bound for the workload in the system, and using the link speed or shaper rate to calculate the service time for that workload. This section provided formal proofs for worst-case delays with all four shapers in decentralized architectures. For SP and CBS, additional Network Calculus models were derived that provide

TABLE IV  
OVERVIEW OF KPIS AND EVALUATION SCENARIOS

| KPI                             | Scope       | Description                                                  |
|---------------------------------|-------------|--------------------------------------------------------------|
| $ S_a^P $                       | priority    | Number of accepted streams                                   |
| $r_a^P$                         | priority    | Sum of data rates of accepted streams                        |
| $b_a^P$                         | priority    | Sum of bursts of accepted streams                            |
| $d_P^{seg}$                     | prio., link | Latency bound for priority $P$ and segment $seg$             |
| $\mathcal{L}_i^+$               | stream      | Accumulated maximum latency of stream $i$                    |
| $\mathcal{L}_i^\Delta$          | stream      | Jitter: difference between max and min latency               |
| $b_i^{eff}$                     | stream      | Effective burstiness (at the receiver device)                |
| $t_{comp}$                      | total       | Computation time for entire scenario                         |
| Scenario                        |             | KPIs                                                         |
| preconf: fixed delay thresholds |             | $ S_a^P , r_a^P, b_a^P, d_P^{seg}$                           |
| autoconf: accepting all streams |             | $\mathcal{L}_i^+, \mathcal{L}_i^\Delta, b_i^{eff}, t_{comp}$ |
| single switch parameter study   |             | $d_P^{seg}$                                                  |

tighter bounds at the expense of more required information and more complex computation. They benefit from the serialization effect: additional workload cannot appear instantaneously, it is received over time, while the previous workload can already be serviced. To bridge the gap between formal theory and applicability, pseudo code implementations are suggested for each discussed model.

Using Network Calculus theorems like *leftover service* or *output characterization* [170, Sec. 2.3] directly with the service curves is avoided here, as that would require frequent recalculations when streams change anywhere in the network. The accumulated maximum latency is used with the Traffic Specification to construct local arrival curves instead.

## VII. EVALUATION

This section presents a comparison of the five different latency models. A summary of the evaluated Key Performance Indicators (KPIs) and scenario parameters is provided in Table IV. The KPIs used to quantify model performance in practice are explained in Section VII-A. The network topologies used for the evaluations are detailed in Section VII-B. The testing methodology and results are then presented for three distinct scenarios in Sections VII-C, VII-D, and VII-E: (VII-C) a pre-configured scenario with fixed delay thresholds where stream acceptance is analyzed, (VII-D) an auto-configured scenario accepting all streams to evaluate end-to-end guarantees, and (VII-E) a single-switch parameter study observing individual parameter effects. These scenarios are also summarized in Table IV, and their implementation is available on GitHub<sup>2</sup>. Finally, a summary and shaper selection recommendations are provided in Section VII-F.

### A. KPIs

Resource reservation for deterministic networking is fundamentally an admission control process, where streams requesting allocation in high-priority queues are either accepted or rejected along their path. Consequently, the most direct performance indicator is the **number of successfully admitted stream reservations**  $|S_a^P|$  in a given scenario. This metric

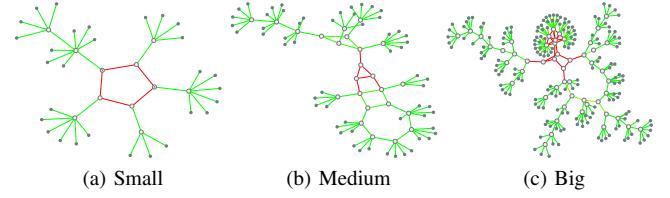


Fig. 16. Randomly sampled industrial scenarios; link colors indicate relative bandwidth (100M-1G in small, 100M-10G in medium and big topologies)

is mapped to underlying traffic characteristics: the **sum of accepted data rates**  $r_a^P$  and **bursts**  $b_a^P$ , since larger traffic volumes demand more resources. Furthermore, the **calculated latency bound**  $d_P^{seg}$  is a direct indicator of these measures, as lower latency predictions permit more admissions.

However, this comparison is not straightforward. Most decentralized latency models presented here require pre-configured latency thresholds  $\delta_P^{seg}$  to function. A latency bound  $d_P^{seg}$  remains valid only if every network link stays below its configured threshold  $\delta_P^{seg}$ . If excessive stream admissions cause  $d_P^{seg}$  to exceed  $\delta_P^{seg}$  at any point, the underlying assumptions of the latency models behind Section VI fail, invalidating all calculated bounds across the network. Consequently, in scenarios where reservations are rejected, the resulting latency bounds  $d_P^{seg}$  cannot properly reflect the performance of the model. They simply reflect the pre-defined configuration rather than actual model performance.

To resolve this issue, the ‘autoconf’ scenarios do not use pre-configured thresholds. Instead, thresholds  $\delta_P^{seg}$  are increased iteratively until they closely match the calculated delays  $d_P^{seg}$  and all streams are accepted. Under this near-optimal configuration, the streams’ performance is represented by the **accumulated maximum latency**  $\mathcal{L}_i^+ = \sum_{seg} \delta_P^{seg}$ , at the receiver. Other bounds may be required depending on the use case: distributed control loops could prioritize an upper bound for **latency jitter**  $\mathcal{L}_i^\Delta = \mathcal{L}_i^+ - \mathcal{L}_i^-$ , while low-tier devices with small buffers may require low **effective burstiness**  $b_i^{eff}$ .

Finally, the total **computation time**  $t_{comp}$  of all latency calculations within a scenario is measured. Note that this is not a sophisticated measurement, it can only provide a rough comparison. The execution time in Python is subject to random effects such as Garbage Collection, and there are no optimizations considered, such as memorizing intermediate results, which can have significant effects as demonstrated in [174, Fig. 6c]. However, the timings still provide a rough overview of the relative complexity of the different algorithms.

### B. Network Topologies

The performance evaluation is conducted on three network topology types: (i) a linear topology representing daisy-chained industrial automation networks, (ii) a set of synthetic industrial topologies with varying sizes, link speeds, stream paths, and Traffic Specifications, and (iii) a single-switch topology with varying amounts of connections for the parameter study.

*Linear Topology:* Since realistic industrial networks are heavily based on linear chains of devices [175, 176] to facilitate factory deployment, a *linear topology* is used to

<sup>2</sup><https://github.com/lsinfo3/tsn-decentralized-network-planning>

establish a baseline scenario without random variance. A chain of 25 switches is modeled, with each switch connected to four end devices. The first switch is connected to a Programmable Logic Controller (PLC) that receives traffic from all 100 devices, resulting in path lengths of 2 to 26 hops. Each device generates a single stream with a burst size of 120 bytes and a data rate of 150 kbits/s, representing periodic industrial control loop traffic, roughly one packet every 6.4 ms.

**Industrial Topologies:** The industrial topologies extend the daisy-chain concept using synthetic networks modeled after common industrial automation patterns. These configurations are inspired by the IEC/IEEE 60802 industrial profile [176] and PROFINET design guidelines [175], which were combined into a scenario generation tool in prior work [177].

The initial topology generator was based on traditional industrial infrastructures. Typically, the field level consists of narrow linear chains of low-port switches representing individual machines or sensor arrays, which connect to the control level via the built-in switch of their PLC. At the control level, switches are often arranged in a ring topology for redundancy, which connects to the plant level via redundant links supporting higher bandwidth. The plant level itself is often structured like the higher layers of a fat tree topology.

The tool from [177] was further extended to anticipate future industrial trends, such as the migration of physical PLC functions to virtual PLCs in plant-level mini-clouds. This shift increases the scale, heterogeneity, and stream count at the plant level. Technologies like TSN enable such real-time communication over Ethernet, beyond the traditional closed systems, which makes these heterogeneous scenarios highly suitable for evaluating TSN latency models.

Example topologies for the small, medium, and big profiles are depicted in Figure 16. A small topology comprises a main chain of 2–5 switches (1 Gbit/s, red), connected to a ring in 50% of the cases, with linear branches of length 1–3 (100 Mbit/s, green) and 3–6 end devices per branch switch. Note that the two main switches in Figure 16a are marked with an ‘X’: they serve as *join points* that can be used to combine multiple topologies. Essentially, the medium scenario is just a combination of multiple small topologies, connected to a 10 Gbit/s main ring; cf. Figure 16b. The big scenario scales this up by connecting a small fat-tree representing a local data center with 1 Gbit/s links to the main ring.

Stream paths, bursts, data rates, and priorities are generated randomly. Short paths are generated more frequently than longer paths. Small, medium, and big topologies contain 110, 220, and 430 streams, respectively. Priorities are selected randomly from 4 to 7, burst sizes are uniformly distributed in the range [64 bytes, 1000 bytes], and data rates are exponentially distributed in the range [10 kbits/s, 50 Mbits/s] to favor lower data rates. All streams utilize shortest-path routing without assuming a spanning tree.

**Parameter Study Topology:** The parameter study network is depicted in Figure 17. It consists of a single switch with several senders transmitting to a single receiver. The number of senders varies dynamically, and the delay analysis focuses on the egress segment *seg* connecting the switch to the receiver.

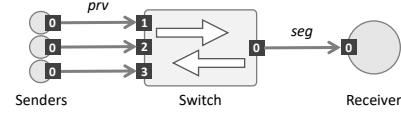


Fig. 17. Single switch evaluation scenario

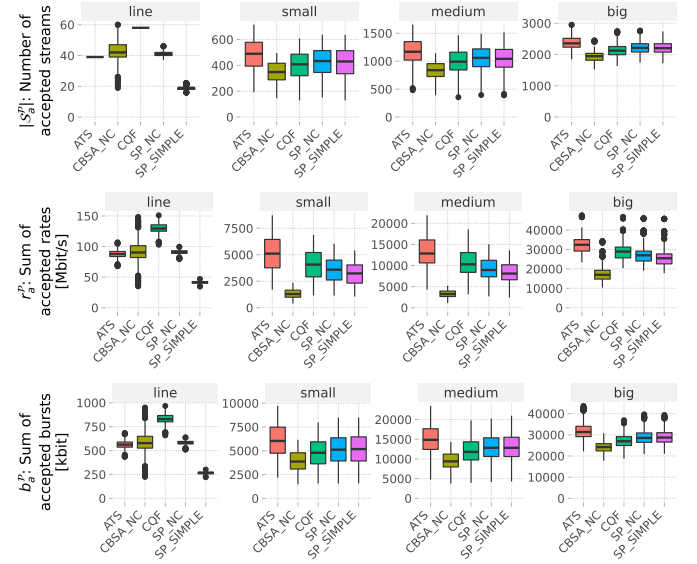


Fig. 18. Comparison of accepted streams, throughputs, and bursts per scenario

### C. Pre-configured Scenario & Accepted Streams

In the initial comparison, the four network sizes (linear, small, medium, and big) are pre-configured with static default latency and bandwidth utilization thresholds without optimizing for individual links. Stream reservations are simulated sequentially. This sequential order creates an implicit stream prioritization: earlier reservation requests are more likely to be accepted, as there are more resources available in the beginning.

This setup can create unexpected bottlenecks and uneven resource utilization, complicating model comparisons. For example, admitting a single high-burst stream with a long path consumes more resources than multiple short, low-burst streams. If one shaper successfully accommodates this larger stream, it may reject subsequent smaller reservations due to exhausted resources. Conversely, a *slightly* less accurate shaper might reject the large stream immediately, allowing it to accept more small streams. Comparing only the total number of accepted reservations  $|S_a^P|$  would incorrectly suggest that the second shaper performed better despite predicting higher latencies.

To mitigate this bias, two actions are taken: (i) the total accepted bursts  $b_a^P$  and rates  $r_a^P$  are evaluated alongside  $|S_a^P|$ , and (ii) the stream list is shuffled 100 times per scenario to perform reservation simulations under varying orders, mitigating the impact of unlucky orders.

Specifically, the evaluation procedure is as follows: For all scenario sizes, 100 generated topologies are analyzed across 100 random stream shuffles for each of the five latency models (SP\_SIMPLE, SP\_NC, CBSA\_NC, CQF, AT5), repeating the following steps:

- 1) Generate the network topology. Generate 10 times the base-

line stream count to minimize random statistical variation in acceptance metrics.

- 2) Set per-hop latency thresholds for every delay segment in the network:  $\delta_4 = 5$  ms,  $\delta_5 = 2$  ms,  $\delta_6 = 1$  ms, and  $\delta_7 = 500$   $\mu$ s. For CQF, priorities are consolidated, and thresholds are configured as:  $\delta_4 = \delta_5 = 3.5$  ms, and  $\delta_6 = \delta_7 = 750$   $\mu$ s. These values are based on industrial *network cycle* times [176] and standard profiles [168].
- 3) For CBSA\_NC, set bandwidth thresholds (and idle slopes) to 20% of the link bandwidth for classes 4–7. For other shapers, stream rates are not limited per class, they are only bounded by the total link bandwidth.
- 4) For each stream, test whether any latency threshold would be exceeded by including that stream. If not, permanently admit the stream. Otherwise, mark that reservation as *failed* and proceed without that stream.
- 5) Record the list of all accepted streams for this iteration.

**Evaluation Results:** Figure 18 summarizes the three KPIs: accepted streams  $|S_a^P|$ , aggregate accepted rates  $r_a^P$ , and aggregate accepted bursts  $b_a^P$ . Each boxplot data point represents a single topology under the specified parameters. The plots are grouped by topology size, and colors indicate the latency models. In addition, Figures 19 and 20 zoom into specific scenarios. For most plots except for Figure 20b, higher values indicate better model performance. Key observations include:

(1) The linear topology behaves differently than the three industrial topologies (Figure 18). Within the industrial subset (small, medium, big), identical trends occur across all three KPIs, indicating that network size has minimal impact on shaper comparison as long as structural properties are maintained. However, the *network type* and shape, which affects path lengths, number of ingress ports, and fan-in/fan-out, can alter the preferred latency model. Specific parameters are analyzed in Section VII-E.

(2) In the linear topology, CQF appears to perform best, but this is most likely an artifact of the configuration. CQF uses a higher latency threshold to compensate for priority consolidation. While this configuration usually reduces stream capacity, the linear network evaluated here contains only priority 7 streams, neutralizing consolidation penalties. The higher threshold directly translates to more accepted streams, which does not occur in other topologies.

(3) In the linear topology, ATS, CBSA\_NC, and SP\_NC show similar performance across all KPIs. In contrast, SP\_SIMPLE accepts only half as many streams, demonstrating the capacity benefits of the other models' improvements.

(4) In industrial topologies, CBSA performs worst across all KPIs, particularly in the rate comparison, indicating high sensitivity to high-rate streams. Although CBSA\_NC is the only model requiring a bandwidth cap (20% per priority), its lower performance is visible long before reaching this limit. As shown in Figure 20a, while outliers are indeed capped at 20 Mbit/s, the 75% quantile of the CBSA box is significantly below that of other shapers.

(5) Generally, ATS accepts the most streams, throughput, and burst in industrial scenarios (Figure 18). CQF, SP\_SIMPLE, and SP\_NC accept similar stream counts, with

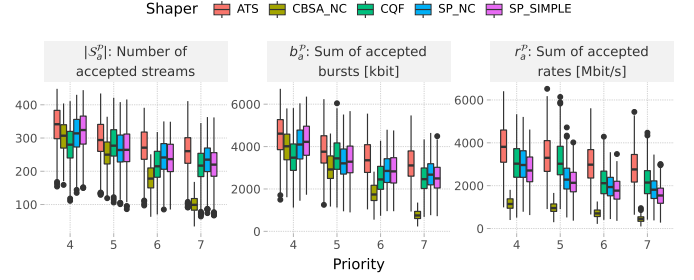


Fig. 19. Comparison of priorities within the industrial medium scenario

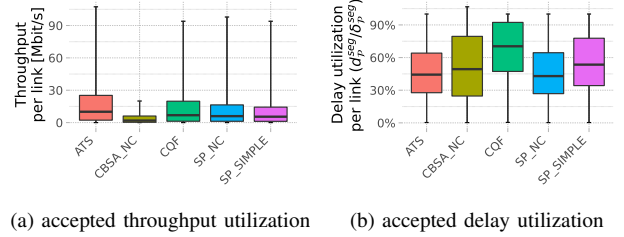


Fig. 20. Accepted utilization per link in the industrial small topology; whiskers extend to the maximum due to an excessive amount of outliers

CQF favoring higher rates and SP favoring higher bursts. In this context, the difference between SP\_SIMPLE and SP\_NC is surprisingly small.

(6) To analyze this further, Figure 19 separates the KPIs in the medium industrial scenario by priority. SP\_NC performs slightly better at high priorities, but the trend reverses at lower priorities. This occurs because admitting more high-priority streams increases lower-priority latencies, consuming their resources. Additionally, in pre-configured scenarios, all links are configured equally, but they are not utilized equally, and a single bottleneck switch can reject many streams. Figure 20b shows the delay utilization distribution of the small industrial scenarios, i.e., the actual delay  $d_P^{seg}$  over the pre-configured threshold  $\delta_P^{seg}$ . Although SP\_NC matched SP\_SIMPLE's overall acceptance in Figure 18, it maintains a lower overall delay utilization. Optimizing thresholds for expected link loads would enable SP\_NC to perform better by using fewer resources. This is evaluated in the auto-configured scenarios.

#### D. Auto-configured Scenario & E2E Guarantees

A drawback of pre-configured scenarios is that static configurations can introduce bias. A specific configuration that works well for one shaper could be disadvantageous for another. Furthermore, uniform link configuration creates highly utilized bottleneck switches while leaving the rest of the network under-utilized. In theory, having an '*optimal*' configuration for each shaper could make their comparison more representative of their real performance, and reveal more differences.

To avoid these pitfalls, an auto-configured study is conducted where all streams are accepted, and per-hop thresholds  $\delta_P^{seg}$  are iteratively matched to calculated delays  $d_P^{seg}$  to eliminate overprovisioning. For all latency models in this paper, the computed delays  $d_P^{seg}$  are non-decreasing when the thresholds  $\delta_P^{seg}$  are increased. Therefore, optimal configurations

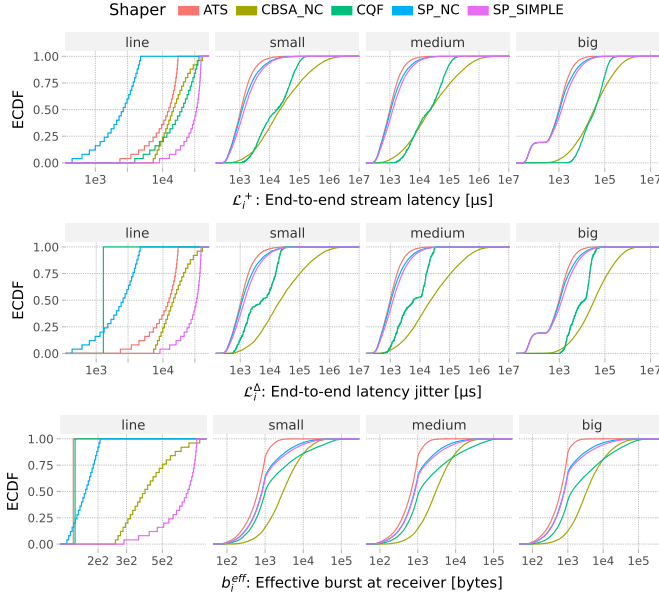


Fig. 21. Comparison of end-to-end latency guarantee, jitter, and effective burst for every topology type

can be achieved by increasing the thresholds  $\delta_P^{seg}$  in small increments until  $d_P^{seg} \leq \delta_P^{seg}$ .

Because all streams are accepted, performance is now measured using end-to-end latency guarantees  $\mathcal{L}_i^+$ , latency jitter  $\mathcal{L}_i^\Delta$ , and effective burstiness  $b_i^{eff}$ , alongside computation time  $t_{comp}$ . For each scenario type and each shaper, the following steps are performed over 100 topologies:

- 1) Generate the network topology and streams as explained before.
- 2) Ensure no link is overutilized, verifying that  $\sum_{i \in S^{seg}} r_i \leq R^{seg}$  on all segments. Remove streams if necessary to satisfy this condition.
- 3) Configure all latency thresholds  $\delta_P^{seg}$  in the network:
  - a) Initialize all thresholds to a default value ( $\delta_{4,5,6,7} = 1$  ns).
  - b) Compute delay bounds  $d_P^{seg}$  based on current thresholds.
  - c) Check if  $d_P^{seg} \leq \delta_P^{seg}$  holds for all segments and priorities.
  - d) If satisfied, terminate the loop.
  - e) If not, update thresholds to  $\delta_P^{seg} = d_P^{seg} \cdot 1.01$  and return to step 3b.
- 4) Record the stream KPIs:  $\mathcal{L}_i^+$ ,  $\mathcal{L}_i^\Delta$ ,  $b_i^{eff}$ , and  $t_{comp}$ .

Note that for CQF, per-link optimization is not possible since the shaper assumes equal hop delays. Step 3e is therefore updated to use the maximum delay:  $\delta_P = \max_{seg} d_P^{seg} \cdot 1.01$ . For CBSA, the idle slope is configured as  $isl_C^{seg} = \sum_{i \in S_C^{seg}} r_i \cdot 1.01$ . Alternative options are explored in Section VII-E.

**Evaluation Results:** Figure 21 presents the Empirical Cumulative Distribution Functions (ECDFs) for each topology size and KPI. Different colors indicate the different latency models. The x-axis is spaced logarithmically, as the results can be several orders of magnitude apart. Generally, the further the ECDF is on the left, the better the latency model performs. Key findings include:

- (1) In industrial topologies, CQF and CBSA show significantly higher latencies ( $\mathcal{L}_i^+$  and  $\mathcal{L}_i^\Delta$ ) than the other shapers,

occasionally by a factor of 100. This difference is much higher compared to the previous pre-configured evaluation. This is expected for CQF, as its global threshold configuration forces the bottleneck link delays to every other link, while all other shapers benefit from the per-hop auto-configuration. For CBSA\_NC, this poor performance stems from a reduced service rate compared to SP\_NC. CBSA can perform much better in general – the conditions for that are explored later in Section VII-E. Here in the industrial scenarios, these conditions are rarely met.

(2) The maximum jitter  $\mathcal{L}_i^\Delta$  is dominated by the maximum latency  $\mathcal{L}_i^+$  since the minimum latency is negligible for most shapers. CQF is the only exception; it ensures that each hop takes exactly  $\delta_P^{seg}$  time, leaving only the last-hop jitter. However, for the industrial scenarios, CQF jitter remains higher than that of locally optimized shapers because its last-hop threshold  $\delta_P^{seg}$  is still quite large in comparison.

(3) ATS performs best across all sizes and KPIs in industrial scenarios, particularly for receiver burstiness  $b_i^{eff}$ . This shows that mitigating the burstiness cascade is an effective strategy – if the switches are actually capable of performing the expensive shaping mechanism. SP\_NC is the second-best option across all KPIs.

(4) Big topologies show a distinct step in latency curves at low values. For ATS and both SP models, roughly 18% of end-to-end latencies fall between 20  $\mu$ s and 100  $\mu$ s. This is an artifact of the topology generation. The big topologies include a small fat-tree section where each end-device is connected with 1 Gbit/s. It is modeled after anticipation of local mini data centers. All the other end devices are connected with 100 Mbit/s. Streams within these high-bandwidth subnets experience shorter paths and lower latency. For CQF and CBSA, this step is not visible. Per-hop optimization is not applicable to CQF, so the delay thresholds are equal everywhere, and CBSA service curves rely on the idle slope rather than the link speed.

(5) Linear topology latency curves ( $\mathcal{L}_i^+$ ) show exactly 25 steps, representing each switch. CBSA and CQF perform better here due to the lack of fan-out. All streams share the same egress port and all delay segment calculations have five ingress ports. SP\_NC outperforms ATS here, likely due to serialization gains. Most streams arrive from the same ingress port, and SP\_NC accounts for sequential arrival over ingress links. With ATS, the packets could theoretically leave the previous shaper simultaneously.

(6) In linear topology jitter metrics  $\mathcal{L}_i^\Delta$ , CQF maintains a constant jitter bound over all streams. CQF's remaining jitter is localized to the last hop, and since all streams share the same last hop, their jitter is fixed by the last hop's threshold  $\delta_7^{seg}$ .

(7) Regarding linear topology burstiness  $b_i^{eff}$ , ATS and CQF provide constant receiver bursts. ATS reshapes traffic directly with the token bucket model, making receiver burstiness dependent on the source Traffic Specifications, independently of their path. CQF reshapes traffic to maintain a constant delay, which directly prevents the burstiness cascading from occurring, as the additional burstiness is based on jitter, cf. Equation 4.

- (8) Between the industrial scenarios, the topology size

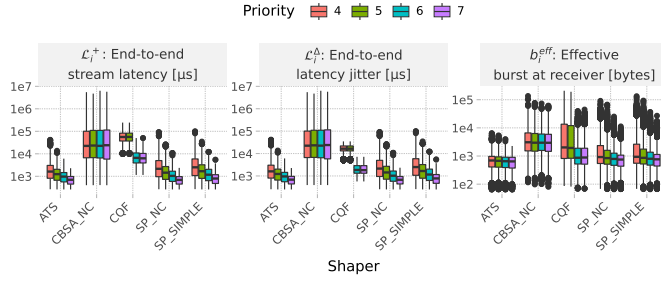


Fig. 22. Influence of the shaper on the relative differences between priorities, zoomed into the medium topology size

appears to have little effect on the relative differences between shapers. Therefore, only the medium topology is analyzed for the remaining observations. Figure 22 breaks down performance by priority. ATS, SP\_NC, and SP\_SIMPLE exhibit delays and jitter that align with priority levels: higher priorities show lower values. This trend disappears under CBSA, likely because the service rate is reduced to the stream data rates, making service rate limits more influential than the presence of higher-priority traffic. Under CQF, the consolidated priority pairs show similar performance. For effective bursts, most observations are similarly applicable, but ATS remains uniform across priorities. Due to per-stream shaping of ATS, the graph reflects the source Traffic Specifications across all priorities.

(9) In the previous figures, the scale of differences is difficult to recognize due to the logarithmic axes. Therefore, Figure 23 focuses on the three close shapers: ATS, SP\_NC and SP\_SIMPLE. The values for ATS and SP\_SIMPLE are normalized to SP\_NC to highlight relative differences. The jitter plot for  $\mathcal{L}_i^\Delta$  was omitted, as it is very similar to the latency  $\mathcal{L}_i^+$ . For burstiness  $b_i^{\text{eff}}$ , ATS consistently matches or outperforms SP\_NC, while SP\_SIMPLE consistently performs worse than SP\_NC. This gap increases at lower priorities, as SP jitter grows. In latency metrics, SP\_NC strictly outperforms SP\_SIMPLE, but ATS and SP\_NC are highly competitive, especially at high priorities. For priority 7, 25% of ATS streams outperform SP\_NC, while 30% of SP\_NC streams outperform ATS. Thus, SP\_NC represents a direct upgrade to SP\_SIMPLE. The choice between ATS and SP\_NC depends on whether per-hop shaping or physical serialization is more dominant in the scenario.

(10) Finally, Figure 24 shows computation times. Each data point of the boxplot represents one scenario, while the lines connect the medians of each boxplot for improved readability. CBSA\_NC is the most demanding model for large topologies, whereas SP\_SIMPLE and ATS are highly efficient. Note that CQF is, in theory, also a simple model, but likely suffers here due to the straightforward implementation of the priority consolidation. In general, sophisticated optimization of the algorithms might shift the timing relations between the shapers, or otherwise reinforce them, but Figure 24 still provides a naive baseline overview of the algorithm complexity.

**Global Configuration Results:** Under global configuration, every link receives the same threshold, which is useful when per-hop optimization is undesired or impossible, e.g., for CQF. This restriction alters several observations from the

TABLE V  
OVERVIEW OF PARAMETER VALUES FOR THE SINGLE SWITCH STUDY

| Parameter               | Default  | Experiment Values            |
|-------------------------|----------|------------------------------|
| bandwidth               | 1 Gbit/s | 100 Mbit/s, ..., 10 Gbit/s   |
| number of ingress ports | 5        | 1, 2, ..., 30                |
| total number of streams | 100      | 1, 2, ..., 1000              |
| total burst             | 32 kB    | 6.4 kB, 12.8 kB, ..., 512 kB |
| total rate percent      | 10%      | 1%, 10%, ..., 75%            |
| acc max delay           | 500 μs   | 0, 50 μs, ..., 10 ms         |
| acc min delay percent   | 50%      | 0, 50%, 99% of max           |
| num high prio streams   | 50       | 0, 50                        |
| idle slope ratio        | 1.0      | 0.1, 0.5, ..., 2.0           |

previous evaluation.

Based on the evaluation procedure from Section VII-D, step 3e is modified as follows:

$$\delta_{\mathcal{P}}^{\text{seg}} = \max_{\text{oseg}} \{d_{\mathcal{P}}^{\text{oseg}}\} \cdot 1.01. \quad (35)$$

Each delay segment *seg* is now configured to the maximum delay  $d_{\mathcal{P}}^{\text{oseg}}$  out of all delay segments *oseg* in the network. Note that this can create circular dependencies: increasing any threshold  $\delta_{\mathcal{P}}^{\text{seg}1}$  will also increase the accumulated latency downstream at other segments *segN*, which will increase their local latency computation  $d_{\mathcal{P}}^{\text{seg}N}$ . If that delay  $d_{\mathcal{P}}^{\text{seg}N}$  now exceeds the previous threshold  $\delta_{\mathcal{P}}^{\text{seg}1}$ , the next auto-config iteration will increase the configuration again, potentially rising indefinitely. For that reason, it is often impossible to obtain an optimal global configuration without rejecting any streams. CBSA\_NC and SP\_SIMPLE suffered from this effect and are omitted here.

Figure 25 shows the global configuration results for medium topologies. ATS outperforms SP\_NC here because SP\_NC burstiness is sensitive to the inflated delay thresholds of global configurations. While ATS end-to-end latency also increases, its burstiness remains bounded by per-hop reshaping. Since all shapers are now restricted to global configuration, CQF achieves the lowest jitter  $\mathcal{L}_i^\Delta$  here. This shows that the latency reduction of CQF is actually effective, although it is usually overshadowed by other effects if optimized per-hop configuration is possible.

### E. Single Switch Parameter Study

The previous evaluations have shown that the performance of the five latency models differ significantly between the different topology types (linear and industrial small/medium/big). In order to isolate the effects and understand them better, this section offers a parameter study with the single switch from Figure 17. A main effects plot is generated for the nine parameters listed in Table V. In each experiment, one parameter is varied while the remaining factors are held at their default values.

The parameters are selected carefully to ensure self-contained scaling. For example, if there was a fixed ‘burst per stream’ parameter, then modifying the number of streams would indirectly modify the total burstiness of each latency equation. The observed behavior would likely be dominated by that increased burst, not by the increased number of streams directly. Thus, the parameters from Table V use *total* amounts, or *relative* definitions based on another parameter.

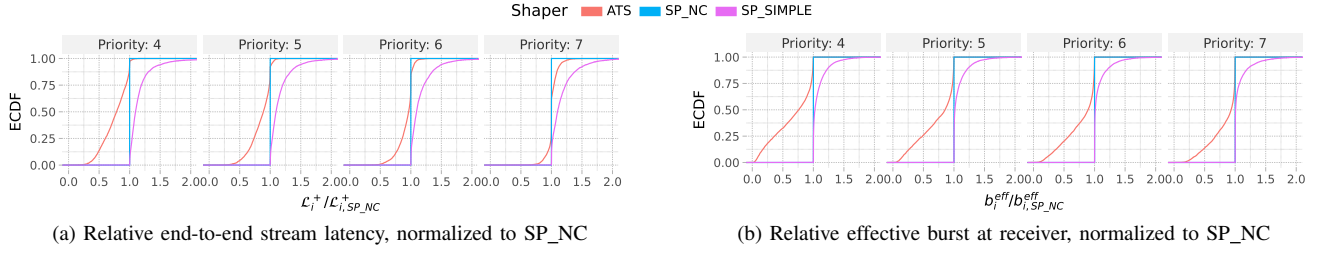


Fig. 23. Relative performance of ATS, SP\_SIMPLE and SP\_NC in the medium topologies

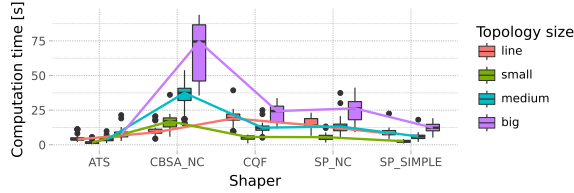


Fig. 24. Required computation time for all latency guarantees

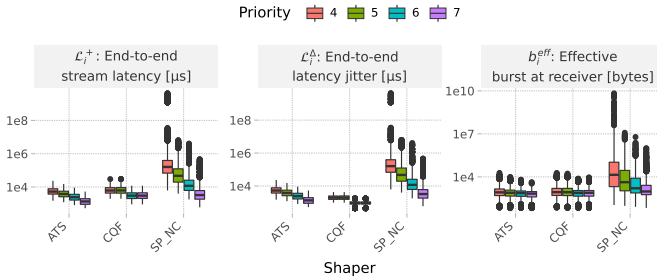


Fig. 25. Global configuration performance of ATS, CQF and SP\_NC in the medium topology

First, the *bandwidth* defines both the bandwidth of ingress links *prv* and the egress link *seg*. The *number of ingress ports* represents the count of upstream links *prv* feeding the switch. More ports increase the number of potentially simultaneous arrivals. The *total number of streams* and *total burst* are distributed evenly across these ingress ports. For example, 10 streams over 3 ports would be distributed as {4, 3, 3}. Individual bursts must still be kept above 64 bytes to comply with minimum Ethernet frame lengths. Therefore, if the number of streams is too high, the effective total burst could be

higher than configured. The *total rate percent* scales relative to link bandwidth. The *accumulated maximum delay* is the worst-case delay that every stream carries up to this point. In the previous full-network evaluations, this value was indirectly given by all per-hop thresholds  $\delta_p^{seg}$  on the paths of the streams. Here, the accumulated maximum delay is controlled directly for a single hop to observe the direct effects. In contrast, the *accumulated minimum delay* is given in percent based on the maximum delay, so it automatically scales. The *number of higher priority streams* controls higher-class interference arriving via a dedicated link. The *idle slope ratio (isl)* dictates CBSA\_NC idle slope parameters. The default slope is equal to the sum of all streams' rates  $\sum_{i \in S^{seg}} r_i$ . For values  $isl \geq 1.0$ , the egress idle slope is multiplied with that value. For values  $isl < 1.0$ , the egress slope cannot be reduced further, as the system would become unstable. Here, the ingress idle slope is divided by *isl* instead, so the parameter always represents the egress/ingress ratio.

**Evaluation Results:** The results of the main effects study are shown in Figure 26. Overall, CBSA\_NC behaves as an outlier, and it performs much better than in the previous evaluations. This came as a surprise and will be covered by the last parameter, the idle slope ratio. Detailed observations are presented in the following.

(1) Higher *bandwidth* reduces latency across all shapers, as the queues are emptied faster.

(2) The *number of ingress ports* has no effect on CQF, ATS, and SP\_SIMPLE, as it is not part of their model. For SP\_NC and CBSA\_NC, more ports increase the immediate burstiness as well as the slope of the arrival curves. SP\_NC converges toward the SP\_SIMPLE bound as the number of ports exceeds the stream count, which matches previous simulation results [178, Fig. 8]. CBSA\_NC has lower arrival slopes because of shaped ingress links, but its service rate is also lowered by egress shaping. After roughly 17 ports, the concurrency penalty outweighs the ingress shaping benefits, and CBSA\_NC delays exceed SP\_NC.

(3) The *number of streams* shows similar effects to the number of ports. The three non-NC models show no effect, except for the last value. For 1000 streams, the individual burst per stream would fall below 64 bytes, which is corrected by the model and shows a higher total delay due to more total burstiness. For SP\_NC, delay increases up to 5 streams, which matches the 5 default ingress ports, as streams can occupy individual ports each. The delay decreases slightly beyond 5 streams because the total burstiness is split between all streams, which takes down the maximum frame lengths,

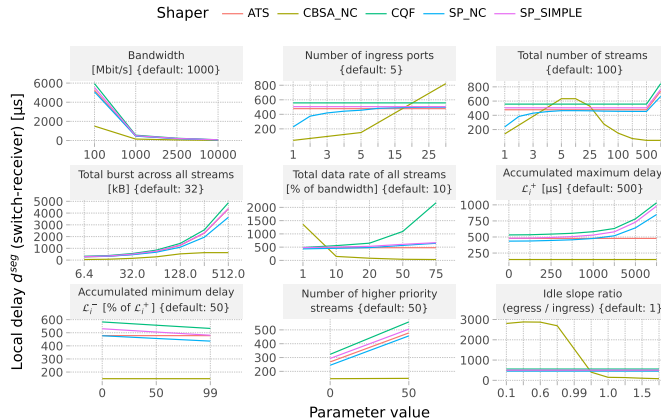


Fig. 26. Single switch parameter study results; x-axis values are categorical and not to scale

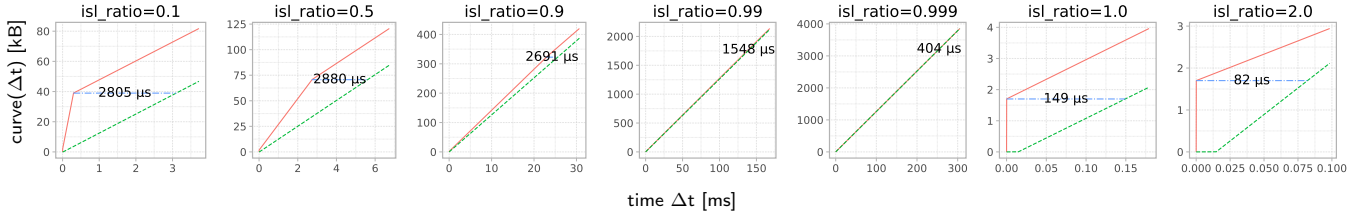


Fig. 27. Overview of network calculus arrival (red) and service curves (green) for different CBSA idle slope ratios

too. CBSA\_NC exhibits the same trend on a larger scale due to rate limitations. Interestingly, the additional burstiness at 1000 streams does not show a significant increase in CBSA's latency – which is consistent with the next observations.

(4) Higher *total burst* increases latency for all models except CBSA\_NC, where delay plateaus after roughly 128 kB. Presumably, at this point, the ratio of burst size  $b_i$  to maximum credits  $maxcr_C^{prv}$  and frame sizes  $\ell_i^{max}$  changes such that the linear piece with rate  $R^{seg}$  from Figure 15a becomes insignificant.

(5) The *total data rate* shows three distinct behaviors. For ATS, the data rate of class  $\mathcal{C}$  has no influence on the delays of the same class, cf. Equation 26. Note that the delay of lower priorities would still be influenced. SP\_SIMPLE, SP\_NC, and CQF show increasing delay with higher rates. CQF exhibits the largest increase, partly due to its consolidated priority interference (Algorithm 5). Partly, it is an artifact of the main effects analysis. In real scenarios, CQF's accumulated minimum latency  $\mathcal{L}_i^-$  would be much higher than the default 50% of  $\mathcal{L}_i^+$  in this evaluation. Since the total data rate is multiplied by jitter  $\mathcal{L}_i^\Delta$  in Algorithm 5, the effect here appears stronger than it would be in practice. Finally, CBSA\_NC latency decreases with higher rates, because the increased egress idle slope dominates the effect of ingress slopes in fan-in scenarios.

(6) and (7) – The *accumulated maximum delay* and *accumulated minimum delay* affect the latency in the same way. Their difference  $\mathcal{L}_i^+ - \mathcal{L}_i^-$  is part of most latency models, except for ATS. Larger maximum jitter increases delays for SP\_NC, SP\_SIMPLE, and CQF. While CBSA\_NC incorporates this jitter, its latency appears to be dominated by other arrival curve segments – at least when the idle slope is given by  $isl_C^{seg} = \sum_{i \in \mathcal{S}_C^{seg}} r_i$ .

(8) *Higher priority streams* increase the delay of the observed class similarly across all shapers except CBSA\_NC. Since CBSA service rate is limited to its idle slope, it remains unaffected. The accumulated positive credit  $maxcr$  and the initial latency of the service curve  $\tau$  could be affected, but the effect appears to be insignificant in this case.

(9) Finally, the *idle slope ratio* reveals an interesting change of order between the shapers. Non-CBS models are unaffected. For  $isl\_ratio \geq 1.0$ , CBSA\_NC delay is smaller than other shapers. However, for  $isl\_ratio < 1.0$ , when ingress idle slopes exceed egress, CBSA delay rises to nearly 6 times the value of other shapers. This ratio appears to be the primary performance factor for CBSA, explaining why it excels in linear configurations but struggles in fan-out scenarios. In fan-out networks, streams diverge across multiple egress ports, reducing egress idle slopes relative to ingress, which degrades

performance in the industrial topologies. This might also explain why CBSA appears to perform much better in shaper comparisons of other papers [81]. This turning point around  $isl\_ratio = 1.0$  can be visualized in the network calculus arrival and service curve plots for CBSA\_NC.

Figure 27 shows snapshots of the arrival curves (red) and services curves (green) of the CBSA Network Calculus model for different idle slope rates. It also shows the maximum horizontal difference between both curves, representing the calculated worst-case delay. For the first five plots of Figure 27 where  $isl\_ratio < 1.0$ , the arrival rate of the second segment  $\alpha_{\mathcal{S},2}(\Delta t)$  is  $isl_C^{prv}$ , which is bigger than the service rate  $isl_C^{seg}$ , so the later bending position  $t_b$  must be selected – cf. Figure 15a. The bigger  $isl\_ratio$  becomes, the smaller  $isl_C^{prv}$  becomes in comparison, until it is almost identical to  $isl_C^{seg}$  for  $isl\_ratio = 0.999$ . For the last two plots, at  $isl\_ratio \geq 1.0$ , it follows  $isl_C^{seg} \geq isl_C^{prv}$  (the green slope is bigger than the red slope), so the first bending position  $t_{cr}$  from Figure 15a is selected – much earlier than before. The delay is now given by the time it takes to service the immediate bursts  $maxcr_C^{prv}$  from the predecessors, and no longer based on the original burstiness of the Traffic Specifications.

These results show that CBS can be a good option if the egress-to-ingress idle slope ratio is maintained above 1.0. This can be achieved via flow steering to avoid fan-out, or by overriding the configuration of idle slopes such that they never become smaller along the path of streams. It strongly depends on the network topology whether this is feasible. If these criteria cannot be met, it should be possible to switch to SP\_NC at any switch on the path without additional delay penalties. Future work could focus on automating these transitions.

## F. Lessons learned

It is difficult to provide generalized recommendations for shapers, their relative performance is highly situational. While using default global configurations showed very little differences between shapers (cf. Fig. 18), optimal per-hop configuration reveals orders of magnitude between them (cf. Fig. 21), and a factor of 2.0 (cf. Fig. 23) between points that previously looked similar. This section demonstrated how different shapers can still be compared fairly, without introducing bias through non-optimal configurations. The following paragraphs summarize the results for the industrial and linear topologies.

First, the improved SP\_NC outperforms SP\_SIMPLE across all scenarios and KPIs, except for computation time. If signaling and algorithmic complexity are acceptable, SP\_NC

should always be preferred (cf. Fig. 23). Especially in global configuration scenarios, SP\_SIMPLE often failed to provide *any* solution that would accept all streams due to cyclic dependencies.

Second, the optimal choice of shaper depends on the target KPI and topology. CBS can outperform other shapers if the egress idle slope exceeds the aggregate ingress slopes (cf. Fig. 26). In scenarios with high fan-out, SP\_NC was always a better choice. Similarly, CQF excels at minimizing jitter under global configurations (cf. Fig. 25), but with local threshold optimization, SP\_NC performed better with respect to latency *and* jitter. Finally, ATS performed well throughout all evaluations as it negates burstiness cascading. It showed the best results with respect to burstiness at the receiver, and it performed similarly to SP\_NC regarding worst-case latency. The serialization effect of SP\_NC can still outperform ATS (cf. Fig. 23, priority 7), especially when there are switches with very few ports. At lower priorities, ATS consistently outperforms SP\_NC (cf. Fig. 23, priority 4). The primary disadvantage of ATS is hardware cost and availability of the shaper.

Overall, for decentralized architectures, SP\_NC is a robust choice across all scenarios. The use of specialized shapers (CBS, CQF, ATS) remains situational, but can be beneficial, and should be verified per use case.

## VIII. DISCUSSION AND FUTURE WORK

While this paper attempts to be self-contained and comprehensive, some caveats must be understood to put the results in full context. This tutorial is centered on the decentralized applicability of the presented concepts. Consequently, the shaper-to-shaper delay segment model (Fig. 5) from the ATS delay analysis was adopted for all mechanisms. While this adaptation enables decentralized CQF applicability, it introduces some distinct tradeoffs.

(1) One major tradeoff is already covered in Table III (Section V). To compute the worst-case queuing latency of the upstream switch, the downstream switch requires additional information that must be signaled during reservation. This requirement complicates the design and standardization of reservation protocols, but it has negligible consequences afterwards.

(2) The delay of a single segment can no longer be measured directly. Although the delay of an entire device can be captured by tapping links before and after that device, it is not so simple to derive the exact time when the shaping step is finished *inside* the device. Verification testbeds would require access to internal component logs, and even then, time stamps across different devices remain difficult to compare without tight time synchronization.

(3) Hybrid deployments and horizontal interoperability are simplified by a shared delay segment definition. For example, SP\_NC can be deployed globally, while a few critical switches utilize ATS to reshape traffic and counter burstiness cascading. Such hybrid topologies offer targeted performance benefits at a lower deployment cost, however, they require theoretical adjustments. First, current latency models still assume homogeneous upstream shaping. The ATS model must be adapted

to include additional shaping delay if the predecessor is non-shaping. Second, current SP and SP\_NC models recompute the effective burstiness  $b_i^{eff}$  at each hop, independently of other shapers along the path. To resolve this, reservation protocols should propagate both the current *effective* Traffic Specification along with the original *talker* Traffic Specification. This allows shapers to signal their changes to the effective burstiness further downstream. Recent RAP drafts already introduce separate Talker and Network TSspecs for this purpose (cf. [7, Sec. 6.3.10]).

(4) Beyond horizontal interoperability, i.e., different shapers used in different delay segments, vertical interoperability should also be considered, i.e., different shapers used by different traffic classes of the same device. While the logic from Section VI can be extended to derive combined class models, as demonstrated in [81], this was omitted here due to the vast combination space. When combined shaper setups are desired, the latency models should be adapted accordingly.

(5) The shaper-to-shaper model is not fully compatible with all common networking mechanisms, including some analyzed in this work. As noted in Section VI-E, CBS is non-compliant because its shaping state transitions occur during frame transmission, which is subject to queuing delays. Consequently, for CBS, the formal *shaping* step includes the queuing and transmission phases, leaving the literal *queuing* step empty (cf. Fig. 14). Unlike the other four models, CBS requires each device to compute *its own* queuing latency. While this fits homogeneous CBS networks, hybrid transitions (e.g., ATS to CBS) require downstream bridges to compute both upstream and downstream queuing delays. Exposing these transition boundaries within the reservation protocol could provide additional flexibility. While CBSA must always compute its own queuing delay downstream, and ATS and CQF always compute the upstream's queuing delay, the SP models are free to be designed either way for full compatibility. In [164, p. 14], we have already suggested to include an indicator in the reservation protocol to specify whether the upstream device has already dealt with its own queuing delay, or left it to the downstream device to consider. This means that the queuing process no longer needs to be priority based. For example, end devices that are not capable of priority transmission selection could employ weighted fair queuing instead. As long as a proven queuing delay bound exists, the upstream device can use it, indicate that it has dealt with the queuing delay, and the downstream switch no longer has to bother.

(6) Special care is required when using reservation protocols with redundancy. In Frame Replication and Elimination for Redundancy (FRER), a sequence number is encoded into the frame by adding a tag, which increases the frame's length and the stream's burst size. Furthermore, even after duplicate elimination, the burst size of the merged stream must often be doubled to consider worst-case events. If maximum latency jitter is high enough on both redundant paths, the frames may experience different delays on both paths. This is usually fine, as the second occurrence of every frame is eliminated anyways, but it creates a problem if the quicker path is temporarily disrupted. After the link is re-established, the merging point receives older frames from the slower link, and

more recent frames from the quicker link. Both of these are valid and will not be eliminated, which temporarily causes a higher burst after merging, until all of the older frames from the slower link are exhausted and the merging point begins to eliminate duplicates again. This is explained in [179] and [6, Sec. C.9] in more detail. Under our framework, ATS could be deployed at the merging point to immediately reshape the traffic, though the combination of FRER and ATS remains subject to ongoing standardization.

(7) This work provides the theoretical foundation for resource reservation, but the ongoing protocol design, configuration, and implementation of resource reservation frameworks are subject of other works. We have conducted complementary studies in some of these aspects. Contributions were submitted to the IEEE 802.1DD RAP standard [163, 164, 180–182], and decentralized reservation performance was evaluated via discrete event simulation [131]. Recently, a proof-of-concept implementation was presented in a demo, including a management API for visualization and root cause analysis [183]. Additionally, this study assumes that stream paths and priorities are known a priori. In practice, assigning streams to priorities is an optimization problem itself. Configuration heuristics exist [127, 184], but they typically assume that all streams are perfectly known a priori. Ideally, a network configuration tool could also work with rough estimates or categories of utilization. A fully decentralized and automated configuration of priority assignments and delay thresholds could significantly simplify the installation process of real time networks.

(8) The latency models in this work do not cover all sources of delay. As categorized in Section IV-A, only transmission and queuing delays are covered. Propagation and internal processing delays must be bounded separately. Decentralized reservation benefits here, as manufacturers can integrate device-specific upper bounds directly into their local protocol stacks. For centralized architectures using a CNC, these bounds must instead be exposed externally, such as through the *digital data sheets* proposed in the IEEE/IEC 60802 industrial profile [1, Sec. 6.4.8.2], which may not fit all custom implementations and can require extensive measurements.

(9) While Time-Aware Shaping (TAS) is widely studied, it has not been covered in this work (except for CQF) due to its reliance on central orchestration. TAS can achieve full frame isolation and zero interference, but it is highly resource-intensive as entire links are reserved for specific streams for a certain time. Consequently, low latency under TAS does not inherently maximize stream density. Future work could evaluate the tradeoffs between asynchronous decentralized concepts and TAS scheduling. Further, both concepts could be combined, e.g., decentralized reservation could bridge the gap between two individual, differently configured TAS domains. This approach could also provide end-to-end guarantees beyond layer 2 networking, across heterogeneous network domains. For example, this has previously been suggested for joint 5G/TSN deployments as an alternative to the logical TSN bridge abstraction [185].

### Lessons learned

The latency models presented in this work lay the groundwork for flexible and decentralized resource reservation. This section pointed out opportunities and requirements for improved heterogeneity. They include additional edge cases in the latency computation, or they add extra information to the reservation protocol. Concrete suggestions include the addition of an effective *Network TSpec* to the original *Talker TSpec*, and an indicator flag identifying whether the upstream device has already accounted for its local queuing delay. Together, these extensions support seamless shaper transitions without sacrificing their individual benefits.

## IX. CONCLUSION

In fields like industrial automation and automotive networks, legacy field bus technologies are slowly replaced by Ethernet to leverage higher bandwidth and standardized support. This transition enables converged networking, where real-time and Best-Effort traffic coexist on a single infrastructure. However, current TSN implementations remain monolithic, similar to the old field bus systems. True interoperability can be achieved with decentralized protocols supported by robust latency models.

This work addresses the challenges of decentralized, heterogeneous deployments with hard real-time guarantees. By formalizing delay segments as the borders of responsibility, shapers like ATS and CQF can be integrated into decentralized architectures. The necessary information flows are identified, including metadata from the upstream devices. Using these foundations, five latency models were derived, ranging from simple formulations (SP\_SIMPLE, ATS, CQF) to Network Calculus implementations (SP\_NC, CBSA\_NC). The models were evaluated using a comprehensive suite of KPIs across linear and industrial topologies.

The evaluations show that priority transmission selection without active shaping remains highly effective. The SP\_NC model consistently ranked among the top-performing approaches. If its computational complexity is acceptable, SP\_NC should be preferred over SP\_SIMPLE. Furthermore, ATS is a highly robust choice for transmission selection, offering simple delay calculations and excellent receiver burstiness bounds. In contrast, CQF and CBSA\_NC are highly situational. When per-hop delay threshold optimization is possible, SP\_NC outperforms CQF, as the benefits of threshold optimization exceed the jitter reduction of global CQF configurations. Finally, CBSA\_NC is effective in topologies with minimal fan-out, where the egress idle slope consistently matches or exceeds the aggregate ingress idle slopes. If the idle slope ratio falls below 1.0 on any link, CBSA\_NC performance degrades rapidly. Because this occurred in almost all tested industrial topologies, SP\_NC is generally recommended instead of CBSA\_NC here. Note that these results highly depend on the topology and stream configuration, but the presented framework can be used to evaluate any given topology.

The contributions in this work are specifically targeted for deterministic delay guarantees in heterogeneous networks. By formalizing the boundaries between different shapers and

clarifying the necessary signaling, this tutorial establishes the mathematical basis for decentralized resource reservation protocols and technological transitions. Future work could branch into several directions, from improved latency models with mixed shapers, to fully self-configuring latency thresholds in decentralized networks. Finally, this work ties directly into the standardization efforts of the IEEE 802.1DD Resource Allocation Protocol and can be used as a direct reference there.

## REFERENCES

- [1] "IEC/IEEE 60802 Time-Sensitive Networking Profile for Industrial Automation," *IEC/IEEE 60802 D3.4*, 2025.
- [2] "IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks – Amendment 31: Stream Reservation Protocol (SRP) Enhancements and Performance Improvements," *IEEE Std 802.1Qcc*, 2018.
- [3] L. Maile, K.-S. Hielscher, and R. German, "Network calculus results for tsn: An introduction," in *2020 Information Communication Technologies Conference (ICTC)*, IEEE, 2020.
- [4] Y. Jiang, "Network calculus bounds for time-sensitive networks: A revisit," *arXiv preprint arXiv:2403.13656*, 2024.
- [5] L. Maile, D. Voitlein, A. Grigorjew, K.-S. J. Hielscher, and R. German, "On the validity of credit-based shaper delay guarantees in decentralized reservation protocols," in *The 31st International Conference on Real-Time Networks and Systems*, ser. RTNS 2023, ACM, 2023.
- [6] "IEEE Standard for Local and Metropolitan Area Networks – Frame Replication and Elimination for Reliability," *IEEE Std 802.1CB*, 2017.
- [7] "IEEE Draft Standard for Local and Metropolitan Area Networks – Resource Allocation Protocol (RAP)," *IEEE Std P802.1DD D1.6*, 2026.
- [8] A. Nasrallah *et al.*, "Ultra-low latency (ull) networks: The ieee tsn and ietf detnet standards and related 5g ull research," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, 2019.
- [9] T. Zhang, G. Wang, C. Xue, J. Wang, M. Nixon, and S. Han, "Time-sensitive networking (tsn) for industrial automation: Current advances and future directions," *ACM Computing Surveys*, vol. 57, no. 2, 2024.
- [10] Y. Peng, B. Shi, T. Jiang, X. Tu, D. Xu, and K. Hua, "A survey on in-vehicle time-sensitive networking," *IEEE Internet of Things Journal*, vol. 10, no. 16, pp. 14375–14396, 2023.
- [11] M. Adil, T. Qiu, X. Zhou, D. Javeed, Z. Cao, and D. O. Wu, "Integrated 5g and time sensitive networking for emerging applications: A survey of advancements, challenges, and future directions," *IEEE Communications Surveys & Tutorials*, vol. 28, pp. 4016–4050, 2025.
- [12] C. Xue, T. Zhang, Y. Zhou, M. Nixon, A. Loveless, and S. Han, "A survey and experimental study of real-time scheduling methods for 802.1 qbv tsn networks," *ACM Computing Surveys*, vol. 58, no. 2, pp. 1–37, 2025.
- [13] A. Nasrallah *et al.*, "Performance comparison of ieee 802.1 tsn time aware shaper (tas) and asynchronous traffic shaper (ats)," *IEEE Access*, vol. 7, 2019.
- [14] D. Yang, Z. Cheng, W. Zhang, H. Zhang, and X. Shen, "Burst-aware time-triggered flow scheduling with enhanced multi-cqf in time-sensitive networks," *IEEE/ACM Transactions on Networking*, vol. 31, no. 6, 2023.
- [15] Z. Pang *et al.*, "Flow scheduling for conflict-free network updates in time-sensitive software-defined networks," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 3, 2021.
- [16] Y. Lu *et al.*, "An intelligent deterministic scheduling method for ultralow latency communication in edge enabled industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 2, 2023.
- [17] X. Wang, H. Yao, T. Mai, S. Guo, and Y. Liu, "Reinforcement learning-based particle swarm optimization for end-to-end traffic scheduling in tsn-5g networks," *IEEE/ACM Transactions on Networking*, vol. 31, no. 6, 2023.
- [18] S. Sudhakaran, K. Montgomery, M. Kashef, D. Cavalcanti, and R. Candell, "Wireless time sensitive networking impact on an industrial collaborative robotic workcell," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 10, 2022.
- [19] M. Guo, C. Gu, S. He, Z. Shi, and J. Chen, "Mss: Exploiting mapping score for cqf start time planning in time-sensitive networking," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 2, 2023.
- [20] Y. Huang, S. Wang, X. Zhang, T. Huang, and Y. Liu, "Flexible cyclic queuing and forwarding for time-sensitive software-defined networks," *IEEE Transactions on Network and Service Management*, vol. 20, no. 1, 2023.
- [21] Y. Zhang, Q. Xu, M. Li, C. Chen, and X. Guan, "Qos-aware mapping and scheduling for virtual network functions in industrial 5g-tsn network," in *2021 IEEE Global Communications Conference (GLOBECOM)*, IEEE, 2021.
- [22] W. Kong, M. Nabi, and K. Goossens, "Run-time recovery and failure analysis of time-triggered traffic in time sensitive networks," *IEEE Access*, vol. 9, 2021.
- [23] Y. Zhang, Q. Xu, C. Chen, M. Li, X. Guan, and T. Q. S. Quek, "Seamless scheduling for nvf-enabled 5g-tsn network: A full-path aoi based method," *IEEE Transactions on Industrial Informatics*, vol. 20, no. 12, 2024.
- [24] I. Alvarez, A. Servera, J. Proenza, M. Ashjaei, and S. Mubeen, "Implementing a first cnc for scheduling and configuring tsn networks," in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE, 2022.
- [25] J. Xue, G. Shou, Y. Liu, and Y. Hu, "Scheduling time-critical traffic with virtual queues in software-defined time-sensitive networking," *IEEE Transactions on Network and Service Management*, vol. 21, no. 1, 2024.
- [26] C. Zhang, X. Wen, C. Li, W. Zheng, L. Yu, and Z. Lu, "Dynamic rapid scheduling algorithm for vehicle time sensitive communication based on cilp and aga," *IEEE Transactions on Vehicular Technology*, 2023.
- [27] D. Brighenti and F. Valenza, "A twofold model for vnf embedding and time-sensitive network flow scheduling," *IEEE Access*, vol. 10, 2022.
- [28] Y. Zhang, C. Chen, Q. Xu, S. Wang, L. Xu, and X. Guan, "Scalable scheduling for industrial time-sensitive networking: A hyper-flow graph-based scheme," *IEEE/ACM Transactions on Networking*, vol. 32, no. 6, 2024.
- [29] L. Zhang, G. Sun, R. Liu, W. Quan, H. Yu, and D. Niyato, "Priority-dominated traffic scheduling enabled ats in time-sensitive networking," *IEEE Transactions on Network and Service Management*, vol. 22, no. 1, 2025.
- [30] J. Lin *et al.*, "Flextas: Flexible gating control for enhanced time-sensitive networking deployment," *IEEE Transactions on Industrial Informatics*, vol. 21, no. 7, 2025.
- [31] Z. Feng, W. Dong, S. Gao, Y. Lin, X. Jin, and Q. Deng, "An efficient heuristic cqf scheduling in time-sensitive networking," *IEEE Transactions on Industrial Informatics*, vol. 21, no. 7, 2025.
- [32] Y. Zhang, Q. Xu, X. Guan, C. Chen, and M. Li, "Wireless/wired integrated transmission for industrial cyber-physical systems: Risk-sensitive co-design of 5g and tsn protocols," *Science China Information Sciences*, vol. 65, no. 1, 2021.
- [33] A. A. Atallah, G. B. Hamad, and O. A. Mohamed, "Routing and scheduling of time-triggered traffic in time-sensitive networks," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 7, 2020.
- [34] V. Gavrilut, L. Zhao, M. L. Raagaard, and P. Pop, "Avb-aware routing and scheduling of time-triggered traffic for tsn," *IEEE Access*, vol. 6, 2018.
- [35] T. Stüber, L. Osswald, S. Lindner, and M. Menth, "A survey of scheduling algorithms for the time-aware shaper in time-sensitive networking (tsn)," *IEEE Access*, vol. 11, 2023.
- [36] Y. Zhang, Q. Xu, L. Xu, C. Chen, and X. Guan, "Efficient flow scheduling for industrial time-sensitive networking: A divisibility theory-based method," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 12, 2022.
- [37] G. Patti, L. L. Bello, and L. Leonardi, "Deadline-aware online scheduling of tsn flows for automotive applications," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 4, 2023.
- [38] X. Jin *et al.*, "Real-time scheduling of massive data in time sensitive networks with a limited number of schedule entries," *IEEE Access*, vol. 8, 2020.
- [39] X. Wang, H. Yao, T. Mai, Z. Xiong, F. Wang, and Y. Liu, "Joint routing and scheduling with cyclic queuing and forwarding for time-sensitive networks," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 3, 2023.
- [40] Q. Yu and M. Gu, "Adaptive group routing and scheduling in multicast time-sensitive networks," *IEEE Access*, vol. 8, 2020.

- [41] Q. Yu, H. Wan, X. Zhao, Y. Gao, and M. Gu, "Online scheduling for dynamic vm migration in multicast time-sensitive networks," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, 2020.
- [42] O. Seijo, J. A. Lopez-Fernandez, and I. Val, "W-sharp: Implementation of a high-performance wireless time-sensitive network for low latency and ultra-low cycle time industrial applications," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 5, 2021.
- [43] C. Zhang *et al.*, "Packet-size aware scheduling algorithms in guard band for time sensitive networking," *CCF Transactions on Networking*, vol. 3, no. 1, 2020.
- [44] W. Sun, Y. Zou, N. Guan, X. Zhang, G. Du, and Y. Wen, "Graph attention network-based deep reinforcement learning scheduling framework for in-vehicle time-sensitive networking," *IEEE Transactions on Industrial Informatics*, vol. 20, no. 7, 2024.
- [45] M. Barzegaran and P. Pop, "Communication scheduling for control performance in tsn-based fog computing platforms," *IEEE Access*, vol. 9, 2021.
- [46] K. Alexandris, P. Pop, and T. Wang, "Configuration and evaluation of multi-cqf shapers in ieee 802.1 time-sensitive networking (tsn)," *IEEE Access*, vol. 10, 2022.
- [47] S. Bhattacharjee, K. Alexandris, E. Hansen, P. Pop, and T. Bauschert, "Latency-aware function placement, routing, and scheduling in tsn-based industrial networks," in *ICC 2022 - IEEE International Conference on Communications*, IEEE, 2022.
- [48] L. Xu, Q. Xu, C. Chen, Y. Zhang, S. Wang, and X. Guan, "Efficient task-network scheduling with task conflict metric in time-sensitive networking," *IEEE Transactions on Industrial Informatics*, vol. 20, no. 2, 2024.
- [49] Y. Zhang, Q. Xu, S. Wang, Y. Chen, L. Xu, and C. Chen, "Scalable no-wait scheduling with flow-aware model conversion in time-sensitive networking," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, IEEE, 2022.
- [50] T. Fiori, F. G. Lavacca, F. Valente, and V. Eramo, "Proposal and investigation of a lite time sensitive networking solution for the support of real time services in space launcher networks," *IEEE Access*, vol. 12, 2024.
- [51] W. Sun, Y. Zou, N. Guan, X. Zhang, J. Fan, and Y. Meng, "Irfis: A cqf scheduling method integrating queue resources and flow features in time-sensitive networking," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 10, 2024.
- [52] B. O. Akram, N. Kamariah Noordin, F. Hashim, M. A. Fadlee Rasid, M. Ismael Salman, and A. M. Abdulghani, "Enhancing reliability of time-triggered traffic in joint scheduling and routing optimization within time-sensitive networks," *IEEE Access*, vol. 12, 2024.
- [53] B. O. Akram, N. K. Noordin, F. Hashim, M. F. A. Rasid, M. I. Salman, and A. M. Abdulghani, "Joint scheduling and routing optimization for deterministic hybrid traffic in time-sensitive networks using constraint programming," *IEEE Access*, vol. 11, 2023.
- [54] Y. Nakayama, R. Yaegashi, A. H. N. Nguyen, and Y. Hara-Azumi, "Real-time reconfiguration of time-aware shaper for ull transmission in dynamic conditions," *IEEE Access*, vol. 9, 2021.
- [55] T. Stüber, M. Eppler, L. Osswald, and M. Menth, "Performance comparison of offline scheduling algorithms for the time-aware shaper (tas)," *IEEE Transactions on Industrial Informatics*, vol. 20, no. 7, 2024.
- [56] W. Tan, B. Wu, S. Wang, and T. Huang, "Large-scale deterministic transmission among ieee 802.1qbv time-sensitive networks," in *ICC 2022 - IEEE International Conference on Communications*, IEEE, 2022.
- [57] X. Li *et al.*, "Determinacy-oriented task offloading scheduling against dos attack for tsn-based edge computing architecture," *IEEE Transactions on Industrial Informatics*, vol. 20, no. 5, 2024.
- [58] X. Hong, Y. Xi, and P. Liu, "Resource-aware online traffic scheduling for time-sensitive networking," *IEEE Transactions on Industrial Informatics*, vol. 20, no. 12, 2024.
- [59] X. Zhou, F. He, and L. Zhao, "Loosely coupled hybrid scheduling of processing and communication for tsn-based ima systems," *IEEE Transactions on Industrial Informatics*, vol. 20, no. 6, 2024.
- [60] Z. Tian, X. Zhou, Z. Liao, M. Sun, and F. He, "Gtsnet: A generalized traffic scheduler for time-sensitive networking based on graph neural network," *IEEE Transactions on Industrial Informatics*, vol. 21, no. 1, 2025.
- [61] A. Arestova, K.-S. J. Hielscher, and R. German, "Optimization of bandwidth utilization and gate control list configuration in 802.1qbv networks," *IEEE Access*, vol. 11, 2023.
- [62] L. Xu *et al.*, "Scalable scheduling in time-sensitive networking: An efficient stream conflict detection method," *IEEE Transactions on Industrial Informatics*, vol. 21, no. 5, 2025.
- [63] Z. Cao, Q. Liu, D. Liu, and Y. Hu, "Enhanced system design and scheduling strategy for switches in time-sensitive networking," *IEEE Access*, vol. 9, 2021.
- [64] H. Chahed and A. Kassler, "Tsn network scheduling—challenges and approaches," *Network*, vol. 3, no. 4, 2023.
- [65] V. Gavrilut and P. Pop, "Scheduling in time sensitive networks (tsn) for mixed-criticality industrial applications," in *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*, IEEE, 2018.
- [66] X. Jin, C. Xia, N. Guan, and P. Zeng, "Joint algorithm of message fragmentation and no-wait scheduling for time-sensitive networks," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 2, 2021.
- [67] N. Reusch, L. Zhao, S. S. Craciunas, and P. Pop, "Window-based schedule synthesis for industrial ieee 802.1qbv tsn networks," in *2020 16th IEEE International Conference on Factory Communication Systems (WFCS)*, IEEE, 2020.
- [68] H.-J. Kim, M.-H. Choi, M.-H. Kim, and S. Lee, "Development of an ethernet-based heuristic time-sensitive networking scheduling algorithm for real-time in-vehicle data transmission," *Electronics*, vol. 10, no. 2, 2021.
- [69] T. L. Mai, N. Navet, and J. Migge, "A hybrid machine learning and schedulability analysis method for the verification of tsn networks," in *2019 15th IEEE International Workshop on Factory Communication Systems (WFCS)*, IEEE, 2019.
- [70] D. Bujosa, M. Ashjaei, A. V. Papadopoulos, T. Nolte, and J. Proenza, "Hermes: Heuristic multi-queue scheduler for tsn time-triggered traffic with zero reception jitter capabilities," in *Proceedings of the 30th International Conference on Real-Time Networks and Systems*, ser. RTNS 2022, ACM, 2022.
- [71] L. Zhao, Y. Yan, and X. Zhou, "Minimum bandwidth reservation for cbs in tsn with real-time qos guarantees," *IEEE Transactions on Industrial Informatics*, vol. 20, no. 4, 2024.
- [72] X. Wang, J. Ren, K. Gong, and H. Zhou, "A delay-optimized routing algorithm for avb streams in time-sensitive networking," in *2022 7th International Conference on Computer and Communication Systems (ICCCS)*, IEEE, 2022.
- [73] J. Prados-Garzon, T. Taleb, and M. Bagaa, "Learnnet: Reinforcement learning based flow scheduling for asynchronous deterministic networks," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, IEEE, 2020.
- [74] L. Maile, K.-S. J. Hielscher, and R. German, "Delay-guaranteeing admission control for time-sensitive networking using the credit-based shaper," *IEEE Open Journal of the Communications Society*, vol. 3, 2022.
- [75] T. Salomon, L. Maile, P. Meyer, F. Korf, and T. C. Schmidt, "Negotiating strict latency limits for dynamic real-time services in vehicular time-sensitive networks," *Vehicular Communications*, vol. 57, 2026.
- [76] J. Miserez, D. Colle, M. Pickavet, and W. Tavernier, "Exploiting queue information for scalable delay-constrained routing in deterministic networks," *IEEE Transactions on Network and Service Management*, vol. 21, no. 5, 2024.
- [77] N. Desai, R. Dobrin, and S. Punnekkat, "Maloc: Building an adaptive scheduling and routing framework for rate-constrained tsn traffic," in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE, 2022.
- [78] A. Grigorjew *et al.*, "Bounded latency with bridge-local stream reservation and strict priority queuing," in *2020 11th International Conference on Network of the Future (NoF)*, IEEE, 2020.
- [79] E. Li, F. He, Q. Li, and H. Xiong, "Bandwidth allocation of stream-reservation traffic in tsn," *IEEE Transactions on Network and Service Management*, vol. 19, no. 1, 2022.
- [80] L. Zhao, P. Pop, and S. S. Craciunas, "Worst-case latency analysis for ieee 802.1qbv time sensitive networks using network calculus," *IEEE Access*, vol. 6, 2018.
- [81] L. Zhao, P. Pop, and S. Steinhorst, "Quantitative performance comparison of various traffic shapers in time-sensitive networking," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, 2022.
- [82] L. Thomas, A. Mifdaoui, and J.-Y. Le Boudec, "Worst-case delay bounds in time-sensitive networks with packet replication and elimination," *IEEE/ACM Transactions on Networking*, vol. 30, no. 6, 2022.
- [83] J. Joung and J. Kwon, "Zero jitter for deterministic networks without time-synchronization," *IEEE Access*, vol. 9, 2021.

- [84] P. Zhang, Y. Liu, J. Shi, Y. Huang, and Y. Zhao, "A feasibility analysis framework of time-sensitive networking using real-time calculus," *IEEE Access*, vol. 7, 2019.
- [85] A. Finzi and A. Mifdaoui, "Worst-case timing analysis of afdx networks with multiple tsn/bls shapers," *IEEE Access*, vol. 8, 2020.
- [86] K. M. Shalghum, N. K. Noordin, A. Sali, and F. Hashim, "Worst-case latency analysis for avb traffic under overlapping-based time-triggered windows in time-sensitive networks," *IEEE Access*, vol. 10, 2022.
- [87] T. Fukui, Y. Sakaue, K. Minami, and T. Taniguchi, "Delay-based shaper with dynamic token bucket algorithm for deterministic networks," *IEEE Access*, vol. 10, 2022.
- [88] E. Mohammadpour and J.-Y. Le Boudec, "Analysis of dampers in time-sensitive networks with non-ideal clocks," *IEEE/ACM Transactions on Networking*, vol. 30, no. 4, 2022.
- [89] M. M. Hasan, H. Feng, S. Khan, M. I. Ullah, M. T. Hasan, and B. Gain, "Improve service curve using non-overlapped gate in time sensitive network switch," in *2021 IEEE 21st International Conference on Communication Technology (ICCT)*, IEEE, 2021.
- [90] J. Specht and S. Samii, "Urgency-based scheduler for time-sensitive switched ethernet networks," in *2016 28th Euromicro Conference on Real-Time Systems (ECRTS)*, IEEE, 2016.
- [91] J. Joung, "Regulating scheduler (rsc): A novel solution for ieee 802.1 time sensitive network (tsn)," *Electronics*, vol. 8, no. 2, 2019.
- [92] D. Maxim and Y.-Q. Song, "Delay analysis of avb traffic in time-sensitive networks (tsn)," in *Proceedings of the 25th International Conference on Real-Time Networks and Systems*, ser. RTNS '17, ACM, 2017.
- [93] F. He, L. Zhao, and E. Li, "Impact analysis of flow shaping in ethernet-avb/tsn and afdx from network calculus and simulation perspective," *Sensors*, vol. 17, no. 5, 2017.
- [94] L. Zhao, P. Pop, Z. Zheng, and Q. Li, "Timing analysis of avb traffic in tsn networks using network calculus," in *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, IEEE, 2018.
- [95] A. E. Kalor, R. Guillaume, J. J. Nielsen, A. Mueller, and P. Popovski, "Network slicing in industry 4.0 applications: Abstraction methods and end-to-end analysis," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 12, 2018.
- [96] L. Zhao, P. Pop, Z. Zheng, H. Daigmore, and M. Boyer, "Latency analysis of multiple classes of avb traffic in tsn with standard credit behavior using network calculus," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 10, 2021.
- [97] E. Mohammadpour, E. Stai, M. Mohiuddin, and J.-Y. Le Boudec, "Latency and backlog bounds in time-sensitive networking with credit based shapers and asynchronous traffic shaping," in *2018 30th International Teletraffic Congress (ITC 30)*, IEEE, 2018.
- [98] L. Zhao, P. Pop, Z. Gong, and B. Fang, "Improving latency analysis for flexible window-based gcl scheduling in tsn networks by integration of consecutive nodes offsets," *IEEE Internet of Things Journal*, vol. 8, no. 7, 2021.
- [99] J. Zhang, L. Chen, T. Wang, and X. Wang, "Analysis of tsn for industrial automation based on network calculus," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE, 2019.
- [100] D. Thiele, J. Schlatter, P. Axer, and R. Ernst, "Formal timing analysis of can-to-ethernet gateway strategies in automotive networks," *Real-Time Systems*, vol. 52, no. 1, 2015.
- [101] B. Zhou, I. Howenstine, S. Limprapaipong, and L. Cheng, "A survey on network calculus tools for network infrastructure in real-time systems," *IEEE Access*, vol. 8, 2020.
- [102] Q. Gao, J. Li, Y. Liu, and Y. Xing, "Optimal control method for congestion control and delay reduction in deterministic networks," *IEEE Access*, vol. 11, 2023.
- [103] E. Li, X. Zhou, J. Sun, H. Xiong, and F. He, "Formal modeling of frame selection in asynchronous tsn communications," *Chinese Journal of Electronics*, vol. 33, no. 2, 2024.
- [104] Y. Yuan, X. Cao, Z. Liu, C. Chen, and X. Guan, "Adaptive priority adjustment scheduling approach with response-time analysis in time-sensitive networks," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 12, 2022.
- [105] X. Zhou, F. He, L. Zhao, and E. Li, "Hybrid scheduling of tasks and messages for tsn-based avionics systems," *IEEE Transactions on Industrial Informatics*, vol. 20, no. 2, 2024.
- [106] K. M. Shalghum, N. K. Noordin, A. Sali, and F. Hashim, "Critical offset optimizations for overlapping-based time-triggered windows in time-sensitive network," *IEEE Access*, vol. 9, 2021.
- [107] X. Dai, S. Zhao, Y. Jiang, X. Jiao, X. S. Hu, and W. Chang, "Fixed-priority scheduling and controller co-design for time-sensitive networks," in *Proceedings of the 39th International Conference on Computer-Aided Design*, ser. ICCAD '20, ACM, 2020.
- [108] I. Val, O. Seijo, R. Torrego, and A. Astarloa, "Ieee 802.1as clock synchronization performance evaluation of an integrated wired-wireless tsn architecture," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 5, 2022.
- [109] O. Seijo, X. Iturbe, and I. Val, "Tackling the challenges of the integration of wired and wireless tsn with a technology proof-of-concept," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 10, 2022.
- [110] S. Bhattacharjee *et al.*, "Network slicing for tsn-based transport networks," *IEEE Access*, vol. 9, 2021.
- [111] V. Balasubramanian, M. Aloqaily, and M. Reisslein, "Fed-tns: Joint failure probability-based federated learning for fault-tolerant time-sensitive networks," *IEEE Transactions on Network and Service Management*, vol. 20, no. 2, 2023.
- [112] G. Miranda, E. Municio, J. Haxhibeqiri, J. Hoebeke, I. Moerman, and J. M. Marquez-Barja, "Enabling time-sensitive network management over multi-domain wired/wi-fi networks," *IEEE Transactions on Network and Service Management*, vol. 20, no. 3, 2023.
- [113] K. Nikhileswar, K. Prabhu, D. Cavalcanti, and A. Regev, "Time-sensitive networking over 5g for industrial control systems," in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE, 2022.
- [114] F. Luque-Schempp, L. Panizo, M.-D.-M. Gallardo, and P. Merino, "Automadapt: Zero touch configuration of 5g qos flows extended for time-sensitive networking," *IEEE Access*, vol. 11, 2023.
- [115] P. Avila-Campos, J. Haxhibeqiri, X. Jiao, B. Van Herbruggen, I. Moerman, and J. Hoebeke, "Unlocking mobility for wi-fi-based wireless time-sensitive networks," *IEEE Access*, vol. 12, 2024.
- [116] N. S. Bulbul and M. Fischer, "Reinforcement learning assisted routing for time sensitive networks," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, vol. 8, IEEE, 2022.
- [117] G. Miranda *et al.*, "Time-sensitive networking experimentation on open testbeds," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, IEEE, 2022.
- [118] A. G. Hagargund, N. S. V. Shet, and M. Kulkarni, "Dtpf algorithm based open-source time-sensitive network leveraging sdn architecture," *IEEE Access*, vol. 11, 2023.
- [119] L. Hu, G. Shou, X. Zhang, Y. Liu, and Y. Hu, "Multi-domain time synchronization model and performance evaluation in tsn," in *2021 7th International Conference on Computer and Communications (ICCC)*, IEEE, 2021.
- [120] G. Miranda *et al.*, "Evaluating time-sensitive networking features on open testbeds," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, IEEE, 2022.
- [121] T. Hackel, P. Meyer, F. Korf, and T. C. Schmidt, "Secure time-sensitive software-defined networking in vehicles," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 1, 2023.
- [122] Y. Zhang, Q. Xu, C. Chen, M. Li, and X. Guan, "Svti: Sfc-driven queue injection in virtual time-sensitive network via augmented topology," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, IEEE, 2022.
- [123] J. Prados-Garzon, T. Taleb, and M. Bagaa, "Optimization of flow allocation in asynchronous deterministic 5g transport networks by leveraging data analytics," *IEEE Transactions on Mobile Computing*, 2021.
- [124] Y. Li, J. Jiang, C. Lee, and S. H. Hong, "Practical implementation of an opc ua tsn communication architecture for a manufacturing system," *IEEE Access*, vol. 8, 2020.
- [125] P. Diederich *et al.*, "Lcdn: Providing network determinism with low-cost switches," in *2025 21st International Conference on Network and Service Management (CNSM)*, IEEE, 2025.
- [126] D. Bujosa, I. Alvarez, and J. Proenza, "Cspr: An enhanced protocol for consistent reservation of resources in avb/tsn," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 5, 2021.
- [127] A. Grigorjew, M. Seufert, N. Wehner, J. Hofmann, and T. Hoßfeld, "Ml-assisted latency assignments in time-sensitive networking," in *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, IEEE, 2021.
- [128] A. Grigorjew, N. Gray, and T. Hossfeld, "Dynamic real-time stream reservation with tas and shared time windows," in *2021 IFIP Networking Conference (IFIP Networking)*, IEEE, 2021.

- [129] A. Grigorjew, C. Baier, F. Metzger, and T. Hoßfeld, "Distributed implementation of deterministic networking in existing non-tsn ethernet switches," in *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, IEEE, 2021.
- [130] I. Alvarez, L. Moutinho, P. Pedreiras, D. Bujosa, J. Proenza, and L. Almeida, "Comparing admission control architectures for real-time ethernet," *IEEE Access*, vol. 8, 2020.
- [131] D. Raunecker, S. Geissler, A. Grigorjew, P. Diederich, W. Kellerer, and T. Hossfeld, "Centralized vs. decentralized: A hybrid performance model of the tsn resource allocation protocol," in *2024 20th International Conference on Network and Service Management (CNSM)*, IEEE, 2024.
- [132] S. Schriegel, T. Kobzan, and J. Jasperneite, "Investigation on a distributed sdn control plane architecture for heterogeneous time sensitive networks," 2018.
- [133] Y. Seol, D. Hyeon, J. Min, M. Kim, and J. Paek, "Timely survey of time-sensitive networking: Past and future directions," *IEEE Access*, vol. 9, 2021.
- [134] A. M. Romanov, F. Gringoli, and A. Sikora, "A precise synchronization method for future wireless tsn networks," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 5, 2021.
- [135] G. P. Sharma *et al.*, "Toward deterministic communications in 6g networks: State of the art, open challenges and the way forward," *IEEE Access*, vol. 11, 2023.
- [136] J. Sasiain, D. Franco, A. Atutxa, J. Astorga, and E. Jacob, "Toward the integration and convergence between 5g and tsn technologies and architectures for industrial communications: A survey," *IEEE Communications Surveys & Tutorials*, vol. 27, no. 1, 2025.
- [137] Y. Wang, S. Yang, X. Ren, P. Zhao, C. Zhao, and X. Yang, "Indust-edge: A time-sensitive networking enabled edge-cloud collaborative intelligent platform for smart industry," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 4, 2022.
- [138] T. Agarwal, P. Niknejad, M. R. Barzegaran, and L. Vanfretti, "Multi-level time-sensitive networking (tsn) using the data distribution services (dds) for synchronized three-phase measurement data transfer," *IEEE Access*, vol. 7, 2019.
- [139] M. Ulbricht *et al.*, "Tsn-flextest: Flexible tsn measurement testbed," *IEEE Transactions on Network and Service Management*, vol. 21, no. 2, 2024.
- [140] L. C. Mutalemwa and S. Shin, "A classification of the enabling techniques for low latency and reliable communications in 5g and beyond: Ai-enabled edge caching," *IEEE Access*, vol. 8, 2020.
- [141] H.-H. Liu *et al.*, "Improving tsn simulation accuracy in omnet++: A hardware-aligned approach," *IEEE Access*, vol. 12, 2024.
- [142] M. Bosk *et al.*, "Methodology and infrastructure for tsn-based reproducible network experiments," *IEEE Access*, vol. 10, 2022.
- [143] J. Sanchez-Garrido, A. Jurado, L. Medina, R. Rodriguez, E. Ros, and J. Diaz, "Digital electrical substation communications based on deterministic time-sensitive networking over ethernet," *IEEE Access*, vol. 8, 2020.
- [144] J. Walrand, "A concise tutorial on traffic shaping and scheduling in time-sensitive networks," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 3, 2023.
- [145] A. Garbugli, L. Rosa, L. Foschini, A. Corradi, and P. Bellavista, "A framework for tsn-enabled virtual environments for ultra-low latency 5g scenarios," in *ICC 2022 - IEEE International Conference on Communications*, IEEE, 2022.
- [146] D. L. Gomez, G. A. Montoya, C. Lozano-Garzon, and Y. Donoso, "Strategies for assuring low latency, scalability and interoperability in edge computing and tsn networks for critical iiot services," *IEEE Access*, vol. 11, 2023.
- [147] Z. Fernandez, O. Seijo, M. Mendicute, and I. Val, "Analysis and evaluation of a wired/wireless hybrid architecture for distributed control systems with mobility requirements," *IEEE Access*, vol. 7, 2019.
- [148] Y. Guan, L. Zhao, J. Hu, N. Lin, and M. F. Alhamid, "Softwarized industrial deterministic networking based on unmanned aerial vehicles," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, 2021.
- [149] H. Fu, M. Sun, B. He, J. Li, and X. Zhu, "A survey of traffic shaping technology in internet of things," *IEEE Access*, vol. 11, 2023.
- [150] L. Maile, D. Voitlein, K.-S. Hielscher, and R. German, "Ensuring reliable and predictable behavior of ieee 802.1cb frame replication and elimination," in *ICC 2022 - IEEE International Conference on Communications*, IEEE, 2022.
- [151] J. Haxhibeqiri, P. A. Campos, I. Moerman, and J. Hoebeke, "Safety-related applications over wireless time-sensitive networks," in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE, 2022.
- [152] H. Zhu, K. Liu, Y. Yan, H. Zhang, and T. Huang, "Measures to improve the accuracy and reliability of clock synchronization in time-sensitive networking," *IEEE Access*, vol. 8, 2020.
- [153] F. Ihle, S. Lindner, and M. Menth, "P4-psfp: P4-based per-stream filtering and policing for time-sensitive networking," *IEEE Transactions on Network and Service Management*, vol. 21, no. 5, 2024.
- [154] Z. Zhou, M. S. Berger, and Y. Yan, "Mapping tsn traffic scheduling and shaping to fpga-based architecture," *IEEE Access*, vol. 8, 2020.
- [155] A. Ben Ahmed, T. Hirofuchi, and T. Fukai, "Hardware design and evaluation of an fpga-based network switch supporting asynchronous traffic shaping for time sensitive networking," *IEEE Access*, vol. 12, 2024.
- [156] T. Weichlein, S. Zhang, P. Li, and X. Zhang, "Data flow control for network load balancing in ieee time sensitive networks for automation," *IEEE Access*, vol. 11, 2023.
- [157] Z. Zhou, J. Lee, M. S. Berger, S. Park, and Y. Yan, "Simulating tsn traffic scheduling and shaping for future automotive ethernet," *Journal of Communications and Networks*, vol. 23, no. 1, 2021.
- [158] I. Behnke and H. Austad, "Real-time performance of industrial iot communication technologies: A review," *IEEE Internet of Things Journal*, vol. 11, no. 5, 2024.
- [159] F. Luo *et al.*, "Design methodology of automotive time-sensitive network system based on omnet++ simulation system," *Sensors*, vol. 22, no. 12, 2022.
- [160] K. Bilstrup, E. Uhlemann, E. Ström, and U. Bilstrup, "On the ability of the 802.11p mac method and stdma to support real-time vehicle-to-vehicle communication," *EURASIP Journal on Wireless Communications and Networking*, vol. 2009, no. 1, 2009.
- [161] "IEEE Standard for Local and Metropolitan Area Network – Bridges and Bridged Networks," *IEEE Std 802.1Q (Revision of IEEE Std 802.1Q-2014)*, 2018.
- [162] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach*, 6th ed. Pearson, 2013.
- [163] A. Grigorjew, "Measurement Points for Worst-Case Per-Hop Latency Computation," in *IEEE 802.1 Interim Session, May, 2022*. Accessed: Apr. 12, 2023. [Online]. Available: <https://www.ieee802.org/1/files/public/docs2022/dd-grigorjew-measurement-points-0522-v02.pdf>
- [164] A. Grigorjew and F. Chen, "Latency Model and Example Reservation Flow in RAP," in *IEEE 802.1 Meeting, January, 2023*. Accessed: Apr. 12, 2023. [Online]. Available: <https://www.ieee802.org/1/files/public/docs2023/dd-grigorjew-latency-model-0123-v03.pdf>
- [165] 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.
- [166] "IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks – Amendment 14: Stream Reservation Protocol (SRP)," *IEEE Std 802.1Qat*, 2010.
- [167] 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.
- [168] "IEEE Standard for Local and metropolitan area networks – Audio Video Bridging (AVB) Systems," *IEEE Std 802.1BA*, pp. 1–45, 2011.
- [169] C. Boiger, "Class A bridge latency calculations," in *IEEE 802 November Plenary Meeting*, 2010.
- [170] Y. Jiang, Y. Liu, *et al.*, *Stochastic network calculus*. Springer, 2008, vol. 1.
- [171] J.-Y. Le Boudec and P. Thiran, *Network calculus: a theory of deterministic queueing systems for the internet*. Springer Science & Business Media, 2001, vol. 2050.
- [172] A. Van Bemten and W. Kellerer, "Network calculus: A comprehensive guide," *TUM Technical Report*, no. 201603, 2016.
- [173] "IEEE Standard for Local and Metropolitan Area Networks — Bridges and Bridged Networks — Amendment: Asynchronous Traffic Shaping," *IEEE 802.1Qcr*, 2020.
- [174] A. Grigorjew, C. Baier, F. Metzger, and T. Hoßfeld, "Distributed implementation of deterministic networking in existing non-tsn ethernet switches," in *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2021, pp. 1–6.
- [175] P. N. e.V., *PROFINET Design Guideline*, 2022. Accessed: Jan. 14, 2025. [Online]. Available: <https://www.profinet.com/download/profinet-installation-guidelines>
- [176] IEC/IEEE 60802, *TSN Profile for Industrial Automation*, Use Cases v1.3, 2018. [Online]. Available: <https://www.ieee802.org/1/files/public/docs2018/60802-industrial-use-cases-0918-v13.pdf>

- [177] A. Grigorjew, V. Vomhoff, S. Geißler, P. Diederich, T. Hoßfeld, and W. Kellerer, “Developing a library of problem scenarios for time-sensitive and real-time networking,” in *International Conference on Networked Systems 2023 (NetSys)*, Sep. 2023.
- [178] A. Grigorjew et al., “Technical report on bridge-local guaranteed latency with strict priority scheduling,” Online Publication Service of Würzburg University (Würzburg University), Tech. Rep., Jan. 2020.
- [179] A. Grigorjew, S. Geißler, P. Diederich, T. Hoßfeld, and W. Kellerer, “Resilience in time-sensitive networking: An overview and open research directions,” in *2023 13th International Workshop on Resilient Networks Design and Modeling (RNDM)*, 2023, pp. 1–8.
- [180] A. Grigorjew, “Bridge-Local Guaranteed Latency with Strict Priority Scheduling,” in *IEEE 802.1 Interim Session, March*, 2020. Accessed: Apr. 12, 2023. [Online]. Available: <https://www.ieee802.org/1/files/public/docs2020/dd-grigorjew-strict-priority-latency-0320-v02.pdf>
- [181] A. Grigorjew, F. Chen, and J. Specht, “Traffic Specification Types in Qdd,” in *IEEE 802.1 TSN Plenary Meeting, July*, 2021. Accessed: Apr. 12, 2023. [Online]. Available: <https://www.ieee802.org/1/files/public/docs2021/dd-grigorjew-tspec-types-0721-v01.pdf>
- [182] A. Grigorjew, “Measurement Points for Worst-Case Per-Hop Latency Computation,” in *IEEE 802.1 Interim Session, September*, 2023. Accessed: Apr. 12, 2023. [Online]. Available: <https://www.ieee802.org/1/files/public/docs2023/dd-grigorjew-managed-objects-0923-v01.pdf>
- [183] A. Grigorjew, T. Hoßfeld, D. Hock, and F. Lipp, “Tsn reservation framework for legacy apps - signaling, shaping, monitoring, and measuring,” in *Proceedings of the ACM SIGCOMM 2025 Posters and Demos*, ser. SIGCOMM ’25, ACM, 2025.
- [184] L. Maile, K.-S. Hielscher, and R. German, “Combining static and dynamic traffic with delay guarantees in time-sensitive networking,” in *EAI International Conference on Performance Evaluation Methodologies and Tools*, Springer, 2023, pp. 117–132.
- [185] A. Grigorjew, S. Raffeck, S. Geißler, and T. Hoßfeld, “Time-sensitive networking over the air: Combining 5g and tsn,” in *KuVS Fachgespräch - Würzburg Workshop on 6G Networks (WueWoWAS’24)*, 2024.