Dynamic Real-Time Stream Reservation with TAS and Shared Time Windows

Alexej Grigorjew, Nicholas Gray, Tobias Hoßfeld
University of Würzburg, Germany
{alexej.grigorjew | nicholas.gray | tobias.hossfeld}@uni-wuerzburg.de

***Abstract*—Motivated by new use cases such as industrial automation and in-vehicle communication, Time-Sensitive Networking is further improving the layer 2 standards for real-time communication in Ethernet networks. As periodic traffic is a common requirement in such scenarios, the Time Aware Shaper (TAS) is a highly popular mechanism that allows the realization of real-time guarantees in synchronized networks. Most applications in literature suggest to configure the TAS for full stream isolation such that the lowest delays can be achieved. However, this "zero-queuing" strategy requires a high amount of computational resources and time, as the configuration of the gate control lists of each port is a complex task, which is not suitable for scenarios with dynamically changing requirements. This paper presents a novel approach with shared time windows. In combination with stream reservation techniques that are commonly applied for asynchronous mechanisms, the network is able to accept new streams during operation without complex reconfiguration, while still benefiting from some degree of isolation compared to conventional asynchronous shapers. The evaluation shows that, given the right circumstances, the number of accepted real-time streams can be improved significantly.**

I. Introduction

Industrial automation has driven modern facilities to rely on periodic synchronization of states between sensors, actors, and programmable logic controllers (PLCs) for several years. Typically, the transmission of these states has been conducted by means of various types of bus systems, including controller area network (CAN) and Profibus. These bus systems offered cheap and simple means of data transmission, featuring different types of collision avoidance (Profibus) and collision resolution (CAN) methods. As there is typically no queuing involved, the implementation of real-time capabilities is simply a matter of allocating bandwidth for each stream. Due to their simplicity, busses have been adopted by other fields with similar requirements, such as in-vehicle networks connecting large amounts of sensors and actors within modern cars.

As technology advanced, these use cases also required more effective bandwidth, and global collision domains without queuing were no longer sufficient. For this reason, various mechanisms based on Ethernet have been developed to enable reliable data transmission with provable latency bounds, e.g., Profinet and EtherCAT. Providing latency guarantees in queued networks is no trivial matter, hence these individual mechanisms have been developed with very different approaches in mind, making them mutually incompatible. The working groups IEEE Time-Sensitive Networking (TSN) and IETF Deterministic Networking (DetNet) are working on a common set of standards on the lower layers 2 and 3 to mitigate these problems.

ISBN 978-3-903176-39-3

Time Aware Shaping (TAS) is one of these standards, and part of IEEE 802.1Q [1]. It is based on a global synchronization of clocks in each participating device. Based on these clocks, each port of each network device receives its own gate control list (GCL) that dictates which traffic classes are allowed to send at which time. In general, it can be seen as a priority based time division multiple access (TDMA) system. As these GCLs are pre-configured and repeat periodically, they are best suited to define transmission windows for periodically synchronizing applications.

Among the TSN shapers, TAS is often mentioned to achieve the lowest latencies [2], but this also implies that an optimized configuration for each individual transmission window in the entire network must be found. Global optimization requires full knowledge of all streams in the network and a high amount of computational resources and time [2], [3], which makes it suitable only for static scenarios. Dynamic use cases, as often envisioned for Industry 4.0, require a more flexible approach. It should be possible to plug and unplug devices, and therefore add and remove new streams at any time without effect on established streams in the network [4].

Contribution: This paper is addressing this requirement by transmitting periodic traffic in synchronized networks with dynamically changing streams and topologies. It relies on simple, pre-configured global time windows for multiple classes of traffic, and a dynamic stream reservation process that can accept new streams for these classes during operation. Therefore, this work suggests a distributed admission control system for new streams with a global TAS configuration, and briefly evaluates its efficiency compared to other shapers. The results show that shared TAS windows may be a feasible alternative to competing asynchronous mechanisms, depending on the specific scenario.

The remainder of this work is structured as follows. Section II covers important background information and related work. In Section III, the stream reservation process is explained, and an admission control mechanism is suggested that checks whether the new reservation still fits into its time window. A comparison of the efficiency of different distributed reservation strategies including Strict Priority (SP, no shaping), Asynchronous Traffic Shaping (ATS), and TAS is presented in Section IV. Finally, Section V concludes the paper.

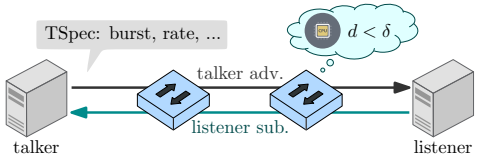


Figure 1. Two step reservation process: talker advertisement with traffic specification, and listener subscription with delay check.

II. Background and Related Work

This section covers important background information that is necessary to understand the concept and is not directly related to timed gates. It includes the aspect of dynamic stream reservation and the computation of latency bounds without perfectly isolating the individual streams through the gates.

A. Distributed Reservation Process

The reservation process, as illustrated in Figure 1, follows a publish-subscribe model. Talkers may indicate that they have information to share by broadcasting a *talker advertisement* in the local network. This advertisement includes the required maximum latency and a traffic specification that specifies how much traffic will be sent in an arbitrary amount of time. This advertisement is received by switches in the network, investigated by their firmware, and flooded to all ports that do not exceed the required latency. Eventually, the advertisement arrives at other end devices, which may in turn subscribe for that data with a *listener subscription* packet. This packet includes the same information as the initial advertisement and travels back to the talker along the previous path. During this process, each switch will once again investigate the packet and calculate the current upper delay bound d for the requested traffic class. If it is still below the pre-configured latency guarantee δ, the new subscription is accepted by the switch and forwarded along the path back to the talker. Finally, the talker acknowledges the subscription to the listener and starts to transmit its data. This procedure is more thoroughly documented in IEEE standards 802.1Qat [5], 802.1Qcc [6] (Stream Reservation Protocol), and 802.1Qdd [7] (Resource Allocation Protocol).

Unlike typical optimization workflows, this process assumes that the switches have already been pre-configured with their latency guarantee δ for each supported traffic class. The precise information of the streams and their traffic volume is only available during operation, after the configuration has already been performed. The reservation process is primarily used to ensure that, from the available pre-configured resources (i.e., bandwidth and delay), a sufficient amount is still available to support the new stream. The amount of additional worst case latency that is inferred from including the new stream is computed based on a distributed latency bound formula, which is presented in the following section.

B. Distributed Worst Case Latency Computation

Given a pre-configured network with dynamic stream reservation, streams cannot be fully isolated by timed gates as often pursued. Instead, the impact of new streams on the existing stream reservations is calculated, and an admission control algorithm compares the currently computed latency bound with the pre-configured guarantee, similarly to asynchronous mechanisms. Some degree of isolation can still be achieved, i.e., groups of streams can be established that only influence each other within the same time window, reducing the overall latency impact. The worst case latency bound within such a group can still be computed based on the same model and formula as the overall latency. This model is based on our previous work [8] and is briefly explained here.

Given the link speed r and traffic specification of each stream x, namely the burst size b_x, the minimum interval between two bursts (period) τ_x, and the maximum frame length $\hat{\ell}_x$, then the worst case upper bound for the per-hop queuing and transmission delay of traffic class p is given by:

$$d_p \leq \sum_{x \in \mathcal{H}_p} \frac{y_p(x) b_x}{r} + \sum_{x \in \mathcal{E}_p} \frac{z(x) b_x}{r} + \max_{x \in \mathcal{L}_p} \frac{\hat{\ell}_x}{r} \tag{1}$$

$$y_p(x) = \lceil (accMaxD_x - accMinD_x + \delta_p)/\tau_x \rceil \tag{2}$$

$$z(x) = \lceil (accMaxD_x - accMinD_x)/\tau_x \rceil \tag{3}$$

The sets $\mathcal{H}_p$, $\mathcal{E}_p$, $\mathcal{L}_p$ contain the streams from higher priorities, the same priority p, and lower priorities respectively. The functions $y_p(x)$ and $z(x)$ indicate the number of bursts from higher priority and equal priority streams x that cause interference and increase the queuing delay of class p. Here, δ_p represents the pre-configured latency bound of traffic class p, and the accumulated latency fields ($accMaxD_x$, $accMinD_x$) are distributed along with the traffic specification during resource reservation. The maximum latency is the sum of all per-hop guarantees δ_p along the path of the stream, while the minimum latency is based on the minimum frame size $\check{\ell}_x$. For more details and a proof of this worst case upper bound, refer to [8].

In this work, the general upper bound is adapted to work with the synchronized TAS windows and to take its potential for isolation into account.

C. Related Work

In general, related work regarding this paper is divided into two categories. The first deals with global optimization of static networks by careful configuration of all gate control lists to achieve maximum isolation of streams, effectively removing all interference. This is pursued with various optimization techniques, such as integer linear programming [3], [9], [10] and satisfiability modulo theories solvers [11]. In general, this category can be extended to include scheduling approaches for other TDMA technologies, such as Profinet and FlexRay [12]–[15]. In addition, some works consider heuristics to find solutions for larger networks, and to adapt to changing network conditions and requirements during operation [16], [17]. In particular, the evaluation of [16] is based on observed statistics of simulated scenarios rather than proven guarantees. This indicates different optimization objectives in literature, as most scheduling approaches are focused on providing the lowest guarantees, or scheduling the largest number of streams. The feasibility of finding valid schedules is mostly limited

by computational resources and, in the case of fully isolated schedules, by available temporal resources to fit all time windows.

The second category investigates the asynchronous computation of worst case delays for a given situation, effectively providing an upper bound guarantee. Unlike explicitly scheduling the traffic classes to isolate individual streams, these approaches do not actively alter the transmission behavior of their underlying shaper with their configuration, but calculate the introduced additional delay instead. This is more suitable for dynamic situations, as it is often less expensive to compute this bound, while also being more bandwidth efficient, since links are not exclusively opened or closed for individual streams. However, the guarantees achieved by this method are inherently not as strict as the scheduled alternative, especially with respect to lower bounds and jitter.

These asynchronous approaches can be further split into those that require full network-wide knowledge, and distributed approaches that can work with local information on each switch. In both cases, the impact of one stream on the latency guarantee of the other streams depends on the distance it has already covered, since higher jitter also leads to a higher potential for accumulating bursts [8]. Therefore, most existing works in literature focus on a static computation with network-wide knowledge of all streams, as this allows to take all such effects into account. Examples include holistic worst case calculations for AFDX [18] and AVB [19] networks, as well as several applications of network calculus [20]–[23]. Distributed per-hop latency computation with switch-local knowledge is often challenging, as changes in the latency of one stream are usually not propagated through the entire network, but they are necessary to adjust the upper bounds of other streams. As shown in Section II-B, this can be approached by using pre-configured guarantees as upper bounds for the varying latency on previous hops. If the same guarantee is being enforced by the admission control algorithm, overestimation is mitigated considerably [8]. Another approach is to prevent the buildup of accumulated bursts in the first place, as pursued by Asynchronous Traffic Shaping (ATS) [24], [25]. The per-hop latency bound of ATS can be solely based on the end devices' original traffic specifications, which are shared with all participating switches during reservation.

In general, work on distributed switch-local computation is very limited. Therefore, to support dynamic use cases, this work presents a switch-local upper latency bound for a pre-configured TAS network with little complexity. It combines asynchronous post-configuration admission control with some degree of isolation (i.e., based on priority) provided by the time aware shaper. This makes it suitable for dynamic scenarios with periodic streams, while still accepting more streams than fully asynchronous approaches.

III. Shared GCLs in Dynamic Networks

Typically, TAS is used to provide full traffic isolation by configuring the GCL of each port individually such that there is only one frame in each queue at any given time. This strategy requires solving a complex scheduling problem and is often very bandwidth inefficient, as some ports must often keep all gates closed to maintain frame isolation on the next hop. Instead, this paper suggests to configure all GCLs in each port equally, implying a single global configuration that is copied to each GCL on each port of the network. This has several implications:

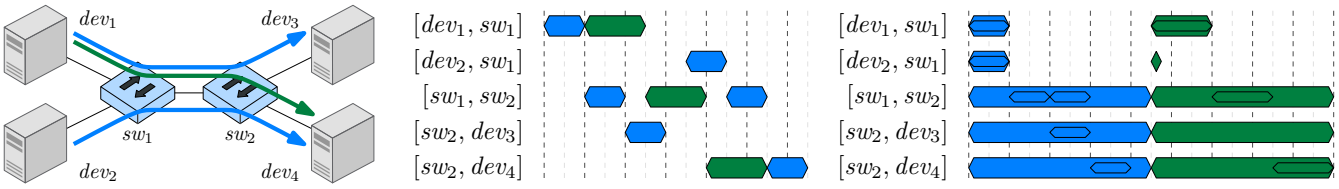
(i) Finding a suitable scheduling for the network becomes much more simple, as there are much fewer decisions to be made. (ii) Frames are no longer fully isolated during transmission, as multiple open gates send towards the same destination at the same time. Multiple frames can be in the queue at the same time. (iii) Time windows in GCLs are now configured for end-to-end transmission instead of per-hop transmission. The window size corresponds to the maximum accepted end-to-end delay during reservation. When a time window ends, all frames from that group must have arrived at their destination. (iv) GCLs do not have to be reconfigured when new streams are deployed in the network. The global time windows are shared by all streams of a given priority. If there is still room within that time window, new streams can simply start to transmit after reservation. (v) As streams no longer have exclusive time windows, they cause additional latency for other streams in the same window. This additional latency must be computed during reservation, and it must be ensured that it does not exceed a previously configured maximum latency during the reservation process. This way, the upper latency bound remains deterministic. This computation is addressed in Section III-B.

A comparison of typical TAS configurations and the suggested global configuration is presented in Figure 2. The example topology considers three streams with two different priorities. In a typical configuration, each time window represents the transmission of a single, individual frame from each stream. In contrast, the global configuration keeps all gates open until each frame has arrived at its destination. Note that this is only true for the configuration of switches. End devices are only allowed to send at the very beginning of each time window, to prevent new frames from being injected into the network when the window is almost expired.

A. Parameters and Configuration

In general, gate control operations [1, Table 8-4] consist of the desired state (open, closed) of the gates for each traffic class, and a time interval after which the next gate control operation is executed. The GCL of each port consists of an ordered list of such gate control operations. After the last operation has finished, the process repeats with the first iteration.

Up to eight individual traffic classes can be supported by a switch [1]. Without loss of generality, it is assumed that up to four traffic classes are reserved for real-time transmissions, and the other four represent different classes of Best-Effort traffic. For each real-time traffic class, three major parameters must be selected. (i) The frequency (or period) of each class must be defined. This refers to the number of gate-open events in



(a) Example topology with 2 switches, 4 devices, and 3 streams.

(b) Typical TAS configuration in literature.

(c) Global TAS configuration for dynamic scenarios.

Figure 2. Parameter description by means of a small example. Blue streams have priority 7, green streams have priority 6.

a given amount of time, and to the interval between two open time windows. (ii) The duration of the open time windows must be defined. This duration is equivalent to the maximum end-to-end delay of the given traffic class. (iii) The per-hop delay of the given class must be defined at each hop. It must be smaller than the entire time window, as the sum of all per-hop delays must not exceed the end-to-end delay. A pre-defined per-hop delay is required for the dynamic admission control during stream reservation. Typically, this delay would be selected based on the expected load. Thus, trunk links between two switches would be configured with bigger delays, while links towards end devices receive stricter guarantees.

For example, in Figure 2c, both windows are open for $4.5\,\text{ticks}$. Both windows are configured to be large enough for the entire end-to-end transmissions of all streams. In real scenarios, these windows would be configured larger to allow new stream reservations during network operation. The per-hop delay guarantees are $1\,\text{tick}$ for blue streams (priority 7) at end devices, $2\,\text{ticks}$ for blue streams between the switches, and $1.5\,\text{ticks}$ for green streams (priority 6) at each port.

Typically, these parameters can be selected by looking at the scenario's requirements. Streams with similar requirements can be grouped a priori, and each group can be assigned to a traffic class. Then, the minimum period of a group defines the frequency of its class, the minimum required end-to-end latency of the group indicates the window size, and the maximum number of hops indicates the fraction of the window size that is used as pre-defined per-hop delay.

Note that, sometimes it may be necessary to open gates for two priorities at the same time, for example if the window size of one class exceeds the period of another class. In this case, streams of both classes are no longer isolated from each other and are part of the same group during the computation of the latency bound for admission control.

B. Distributed Latency Computation with TAS

Similarly to typical TAS configurations, it is assumed that end devices are also synchronized and are able to adapt their transmission behavior to the global network cycle. The delay of a frame is only considered from the moment it is being transmitted, not when the actual data in the end device emerges, i.e., the time from "data available" to "time window begins" is not considered. In addition, due to overlapping schedules, time windows cannot always be reopened periodically after exactly one period, but must sometimes be slightly shifted. This is in accordance with most other approaches, but it should be kept in mind during practical application.

Based on the latency model from Section II-B, switches with up to eight individual priorities and corresponding FIFO queues are considered. The individual streams are now divided into different groups by defining per-priority global time windows. Therefore, the sets $\mathcal{H}_p$, $\mathcal{E}_p$, $\mathcal{L}_p$ from Equation 1 contain fewer streams and the latency bound is improved. In the best case, if time windows do not overlap and guard bands after Best-Effort time windows exist, the only set that must be considered is $\mathcal{E}_p$ containing equal-priority streams: $d_p \leq \sum_{x \in \mathcal{E}_p} b_x / r$. If there are no such guard bands, lower priority transmissions can generally be considered by the time it takes to transmit a single largest frame, including overhead: $\hat{\ell}_{MTU} = 1542\,\text{bytes}$. If time windows do overlap, different priorities must be considered as presented in Equation 1. However, only such priorities that actually do overlap with the observed class p are relevant. These priorities are later referred to as *latency group*. Note that Equation 1 only covers queuing and transmission delays. During productive application, propagation delay, processing delay, and inaccuracies due to the precision of time synchronization must also be considered.

During network operation, new stream reservations are issued as described in Section II-A. This reservation includes a requested priority and a requested end-to-end delay. If the end-to-end delay exceeds the window size, it is trimmed to this size. During reservation, at each hop, the new latency bound d_p is calculated according to Equation 1 under consideration of the latency group. This bound d_p is then compared to the configured per-hop guarantee δ_p. If it does not exceed this guarantee, the reservation is forwarded to the next hop. At the destination, the sum of all per-hop delays, which is contained in the *accMaxD* field in the reservation packet, is compared to the requested end-to-end delay and the window size. If every test has passed, the reservation is acknowledged and the device may transmit its frames at the beginning of each respective time window.

IV. Evaluation

The key performance indicator for the configuration of a real-time network is the number of real-time streams that can actually be supported. This configuration includes the used shaping mechanism, the per-hop guarantees, and the time windows for each priority in case of TAS. Therefore, this section presents a brief comparison of two different configurations for various shapers in two topologies, highlighting when a global

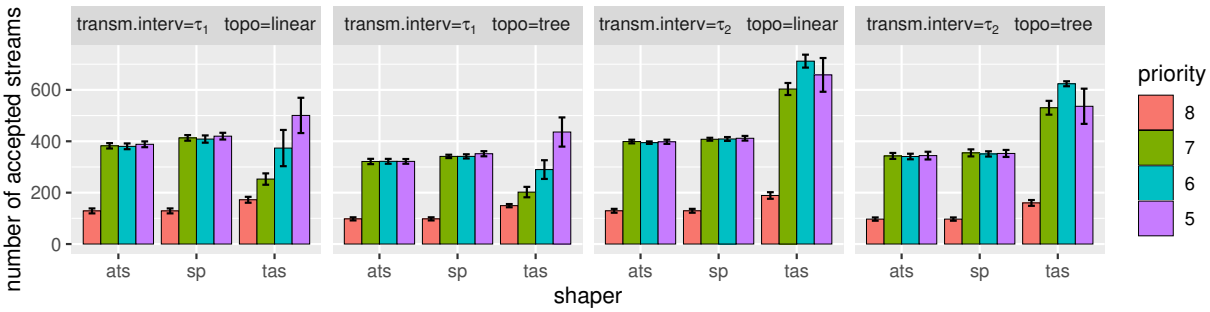


Figure 3. Mean number (and 95% confidence intervals) of accepted streams for various transmission intervals and shapers with distributed stream reservation.

Table I
PROPERTIES OF ALL 4 CLASSES USED FOR THE EVALUATION.

Prio. p	Per-hop delay δ_p	Window size	Period τ_1	Latency group 1	Period τ_2	Latency group 2
8	40 µs	200 µs	1 ms	$\{8,7,6,5\}$	4 ms	$\{8,5\}$
7	200 µs	1 ms	5 ms	$\{7,8,5\}$	10 ms	$\{7\}$
6	400 µs	2 ms	10 ms	$\{6,8\}$	20 ms	$\{6\}$
5	800 µs	4 ms	50 ms	$\{5,8,7\}$	80 ms	$\{5,8\}$

TAS configuration with shared time windows may actually outperform competing asynchronous methods.

Two types of topologies are considered here. The first is a linear topology consisting of 7 consecutive switches in a row. Each switch is connected to 5 end devices, each link has a bandwidth of $1\,\mathrm{Gbit/s}$. The second topology is a binary tree with a depth of 3 and 7 switches in total. On the leaves, all 4 switches are connected to 5 end devices. For both topologies, streams with random properties perform reservations in a simulated environment, and all accepted streams are recorded. The simulation continues until 10 streams of each traffic class have been rejected, as soon as they exceed the latency guarantee and all resources are exhausted. The total number of accepted streams is reported as performance indicator.

The generated streams belong to one of four traffic classes $\{8,7,6,5\}$. Each stream has a random source and random destination node in the entire network. To keep the configuration simple, the maximum hop count is limited to 6. The key properties of each traffic class are reported in Table I. For example, each stream with priority 8 has a per-hop delay guarantee of $40\,\mu\mathrm{s}$ for all switch ports, and an end-to-end delay guarantee of $200\,\mu\mathrm{s}$. This end-to-end guarantee is also configured as the window size for this traffic class. Each class has two different configurations for the transmission interval (period) τ. In the first configuration, all four classes send their frames more frequently, e.g., the first period configuration of traffic class 6 is $10\,\mathrm{ms}$. In the second configuration, periods are higher, therefore frames are transmitted less frequently, and there is more temporal space between two consecutive time windows of the same priority.

The available temporal space between two consecutive time windows is an important information for the GCL configuration. When the time between two windows is not sufficient to schedule all other time windows that are required in that context, some time windows must overlap in order to receive a valid schedule. For example, it is impossible to place a time window for priority 5 with duration $4\,\mathrm{ms}$ between two consecutive time windows of priority 8 with period $1\,\mathrm{ms}$, therefore these two priorities must share a time window and are part of each other's *latency group*. Details of finding a valid schedule are omitted here, as the optimization of this configuration is beyond the scope of this paper. However, it is necessary to know the overlapping groups of priorities in order to compute their latency bound correctly. Table I includes the pre-computed latency groups of all traffic classes based on a simple greedy algorithm. In the second configuration where periods are higher, these groups are smaller. In fact, only classes 8 and 5 overlap in the second configuration, while classes 7 and 6 can be fully isolated, respectively. It is expected that the second configuration is more suitable for the shared TAS mechanism.

Three shapers are compared: shared TAS, reservation with Asynchronous Traffic Shaping (ATS, Equation 21 from [24]), and reservation with Strict Priority (SP) transmission selection (Equation 1, based on [8]). The results of the performed stream reservation simulations are reported in Figure 3. Both types of topologies behaved very similarly, therefore they are discussed together in the following. In general, priority 8 accepted the lowest number of streams due to very strict latency requirements. In addition, ATS and SP performed very similarly because bursts do not accumulate if the delays are smaller than the periods. Note that ATS has previously been shown to perform significantly better than SP if end-to-end delays are bigger than the periods [8]. Here, delays must be smaller to achieve feasible time window configurations.

In configuration 1, TAS was able to accept more streams from priority 8 than SP and ATS. In particular, TAS benefits from guard bands that protect real-time traffic from lower priority Best-Effort transmissions. The additional priority 8 streams have a bigger impact on the other latency bounds than the lower priority Best-Effort traffic would have. As a result, the amount of accepted streams from other priorities in the same group is significantly reduced, as more streams from the highest priority strongly affect their latency bounds. If this behavior is not desired, further measures must be applied that

prevent over-reservation by individual traffic classes.

In configuration 2 with period τ_2, the results for SP and ATS are very similar. Due to slightly lowered bandwidth requirements induced by higher periods, both mechanisms were able to accept slightly more streams in general. However, due to the fully isolated time windows of priority 6 and 7, TAS has been able to accept significantly more streams in each traffic class, both compared to configuration 1 and to the other shaping mechanisms. This shows that using timed gates in conjunction with asynchronous reservation techniques can improve the efficiency of real-time networks with dynamic use cases, given that the scenario and transmission periods allow an efficient isolation of different classes of traffic.

V. Conclusion

This paper presented a novel approach to use the Time Aware Shaper from Time-Sensitive Networking standards for dynamic use cases. Unlike traditional approaches that focus on full stream isolation for the lowest latencies, the presented shared time window mechanism is able to adapt to dynamic changes in streams and requirements without the need for reconfiguration. Therefore, it does not require complex and time-consuming optimization of gate control lists in the entire network. Instead, it is based on a stream reservation process similarly to asynchronous technologies that allows new streams to be introduced during network operation.

The evaluation shows that, given the right circumstances, timed gates can still isolate groups of traffic even when shared time windows are used. Compared to other mechanisms suited for dynamic stream reservation, TAS could accept significantly more streams if the transmission periods were large compared to the window sizes. In such scenarios, TAS presents itself as a feasible alternative to the asynchronous shapers.

There are still possibilities for improvement in the current mechanism. The latency bound can be improved further when more information is shared with the control plane and higher computational capacity is available. The configuration of time windows and per-hop delays can be optimized for specific types of traffic patterns. Finally, GCL configurations could be slightly shifted along a non-circular path in the network to allow higher end-to-end delays without changing the actual window size on an individual switch. These optimizations are subject of future work.

References

[1] "IEEE Standard for Local and Metropolitan Area Network – Bridges and Bridged Networks", *IEEE Std 802.1Q (Revision of IEEE Std 802.1Q-2014)*, 2018.

[2] A. Nasrallah, A. S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. ElBakoury, "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, pp. 88–145, 2018.

[3] S. S. Craciunas, R. S. Oliver, M. Chmelik, and W. Steiner, "Scheduling real-time communication in ieee 802.1 qbv time sensitive networks", in *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, 2016, pp. 183–192.

[4] J. Dorr, *Requirements IEC/IEEE 60802*, Contributions to IEC/IEEE 60802 TSN Profile for Industrial Automation, 2018. [Online]. Available: https://www.ieee802.org/1/files/public/docs2018/60802-industrial-requirements-1218-v12.pdf.

[5] "IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks – Amendment 14: Stream Reservation Protocol (SRP)", *IEEE Std 802.1Qat*, 2010.

[6] "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.

[7] "IEEE Standard for Local and Metropolitan Area Networks – Bridges and Bridged Networks – Amendment: Resource Allocation Protocol (RAP)", *IEEE Std P802.1Qdd*, 2019.

[8] A. Grigorjew, F. Metzger, T. Hoßfeld, J. Specht, F.-J. Götz, F. Chen, and J. Schmitt, "Bounded latency with bridge-local stream reservation and strict priority queuing", in *2020 11th International Conference on Networks of the Future (NoF)*, IEEE, 2020.

[9] E. Schweissguth, P. Danielis, D. Timmermann, H. Parzyjegla, and G. Mühl, "Ilp-based joint routing and scheduling for time-triggered networks", in *Proceedings of the 25th International Conference on Real-Time Networks and Systems*, 2017, pp. 8–17.

[10] N. G. Nayak, F. Durr, and K. Rothermel, "Incremental flow scheduling and routing in time-sensitive software-defined networks", *IEEE Transactions on Industrial Informatics*, vol. 14, no. 5, pp. 2066–2075, May 2018. DOI: 10.1109/tii.2017.2782235.

[11] R. S. Oliver, S. S. Craciunas, and W. Steiner, "Ieee 802.1 qbv gate control list synthesis using array theory encoding", in *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, IEEE, 2018, pp. 13–24.

[12] Z. Hanzalek, P. Burget, and P. Sucha, "Profinet io irt message scheduling with temporal constraints", *IEEE Transactions on Industrial Informatics*, vol. 6, no. 3, pp. 369–380, 2010.

[13] J. Huang, J. O. Blech, A. Raabe, C. Buckl, and A. Knoll, "Static scheduling of a time-triggered network-on-chip based on smt solving", in *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, 2012, pp. 509–514.

[14] P. Pop, P. Eles, and Z. Peng, "Schedulability-driven communication synthesis for time triggered embedded systems", *Real-Time Systems*, vol. 26, no. 3, pp. 297–325, 2004.

[15] H. Zeng, W. Zheng, M. Di Natale, A. Ghosal, P. Giusto, and A. Sangiovanni-Vincentelli, "Scheduling the flexray bus using optimization techniques", in *2009 46th ACM/IEEE Design Automation Conference*, IEEE, 2009, pp. 874–877.

[16] A. Nasrallah, A. S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. Elbakoury, "Performance comparison of ieee 802.1 tsn time aware shaper (tas) and asynchronous traffic shaper (ats)", *IEEE Access*, vol. 7, pp. 44 165–44 181, 2019.

[17] Z. Pang, X. Huang, Z. Li, S. Zhang, Y. Xu, H. Wan, and X. Zhao, "Flow scheduling for conflict-free network updates in time-sensitive software-defined networks", *IEEE Transactions on Industrial Informatics*, vol. 17, no. 3, pp. 1668–1678, 2020.

[18] J. J. Gutiérrez, J. C. Palencia, and M. G. Harbour, "Holistic schedulability analysis for multipacket messages in afdx networks", *Real-Time Systems*, vol. 50, no. 2, pp. 230–269, 2014.

[19] X. Li and L. George, "Deterministic delay analysis of avb switched ethernet networks using an extended trajectory approach", *Real-Time Systems*, vol. 53, no. 1, pp. 121–186, 2017.

[20] K. Lampka, S. Bondorf, and J. Schmitt, "Achieving efficiency without sacrificing model accuracy: Network calculus on compact domains", in *2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, IEEE, 2016, pp. 313–318.

[21] A. Bouillard, "Algorithms and efficiency of network calculus", *École Normale Supérieure*, 2014, Habilitation thesis.

[22] A. Bouillard and É. Thierry, "Tight performance bounds in the worst-case analysis of feed-forward networks", *Discrete Event Dynamic Systems*, vol. 26, no. 3, pp. 383–411, 2016.

[23] S. Bondorf, P. Nikolaus, and J. B. Schmitt, "Quality and cost of deterministic network calculus: Design and evaluation of an accurate and fast analysis", *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 1, p. 16, 2017.

[24] 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.

[25] "IEEE Standard for Local and Metropolitan Area Networks — Bridges and Bridged Networks — Amendment: Asynchronous Traffic Shaping", *IEEE 802.1Qcr*, 2020.