Industrial challenge: Embedded reconfiguration of TSN

Marc Boyer, Rafik Henia

Industrial challenge: Embedded reconfiguration of TSN

Marc Boyer
ONERA/DTIS, Université de Toulouse
31000 Toulouse – France

Rafil Henia
Thales Research & Technology
Palaiseau – France

June 30, 2024

Abstract

This article presents an industrial challenge to the research community.

The sets of Ethernet extensions known as "Time Sensitive Networking" (TSN) is a promising candidate as the next backbone of real-time distributed systems. The flexibility of TSN also an opportunity to reconfigure the network in presence of faults.

This white paper presents an avionic case study for a TSN reconfiguration.

1 Introduction

Redundant communication network architecture plays a crucial role in increasing the fault tolerance of critical real-time communication systems in aircraft. In this setup, multiple redundant network paths are established, ensuring that data can still be transmitted seamlessly even in the event of a failure or disruption in one part of the network. Today, this redundancy in aircrafts is achieved by duplicating AFDX (Avionics Full-Duplex Switched Ethernet) networks. This involves replicating switches, network interfaces, and cabling to ensure uninterrupted communication even in the event of a failure thus maintaining critical avionics systems operational during all stages of flight.

AFDX networks offer increased bandwidth capacity and improved scalability compared to traditional avionics databus protocols. They are designed to meet the stringent requirements of safety-critical avionics applications, such as flight control data, navigation signals, and engine telemetry, providing deterministic communication mechanisms to ensure the integrity and reliability of data exchange.

However, as aviation technology continues to advance, the limitations of the AFDX architecture become increasingly apparent in meeting the evolving communication needs of modern aircraft. One significant limitation is that AFDX networks may struggle to support the increasing complexity and bandwidth requirements of emerging avionics systems, such as those related to autonomous flight, predictive maintenance, and in-flight entertainment. Furthermore, AFDX's deterministic communication model, while crucial for safety-critical applications, may hinder the integration of non-safety-critical systems that require more flexible and dynamic data transmission mechanisms. As the aviation industry seeks to embrace new technologies and enhance passenger experiences, the inflexibility and limitations of AFDX architecture highlight the need for more adaptable and scalable communication solutions to meet the demands of tomorrow's aircraft.

The future deployment of Time-Sensitive Networking (TSN) in aircraft networks represents a significant technological advancement that is expected to substantially improve aviation communication systems. TSN offers several advantages over traditional AFDX networks, including enhanced determinism, scalability, and interoperability. One notable advantage of TSN lies in its capability to support mixed-critical applications within the same network infrastructure. This means that TSN can accommodate a diverse

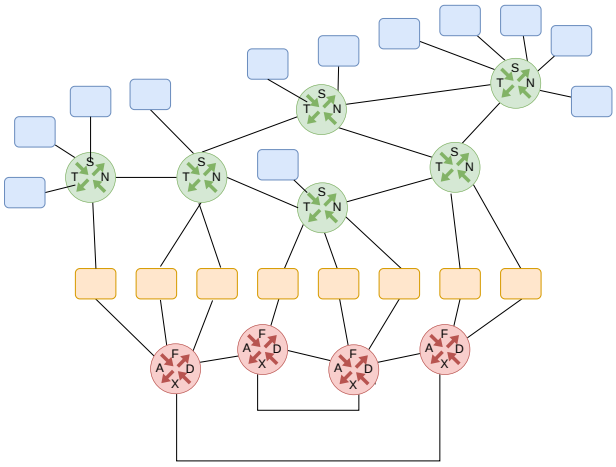

Figure 1: An AFDX-TSN architecture.

range of data traffic with varying levels of importance or urgency, from critical flight control systems to less time-sensitive passenger entertainment systems, all within a single network framework. By leveraging TSN's advanced scheduling and prioritization mechanisms, aircraft manufacturers can streamline network architecture, reduce complexity, and optimize resource utilization while ensuring the stringent safety and reliability requirements of aviation operations are maintained.

The deployment of TSN networks for aviation communication will be gradual over the next years. One can expect the simultaneous deployment of both AFDX and TSN networks in aircrafts. This will offers a strategic advantage in meeting redundancy requirements and reinforcing redundancy asymmetry for avionics networks. While AFDX provides reliability guarantees for critical communications, TSN enhances network efficiency and flexibility particularly in supporting mixed-criticality applications. In such a system, host running critical applications are connected to both networks (as illustrated in Figure 1). In the event of a failure in one of the network, critical applications can still use the other one.

However some hosts are connected to one single network, and one may try to reconfigure the TSN network. Reconfiguring the TSN network allows isolating the malfunctioning components, thus preventing potential cascading failures. Additionally, reconfiguration ensures that the TSN network can be restored once the issue is resolved, thus minimizing the reliance on the redundant AFDX network. Simultaneously, non-critical functions, such as passenger entertainment systems, can also be reestablished, thereby maintaining the overall operational efficiency of the aircraft. This proactive approach to network management is essential for maintaining uninterrupted service and upholding the stringent safety and reliability standards of aviation operations, even in the face of unforeseen failures.

2 Re-configuration challenge

The subject of this challenge is the reconfiguration of the TSN network. More details on TSN, the issues in computing and deploying a new configuration will be presented in next sections. Here are presented some global constraints and objectives:

- *online computation*: One objective is to be able to get the best of the network whatever the number of faults. One approach would be to consider offline all possible faults, and to compute a configuration adapted for each fault. This may hold for small systems or when considering single faults. In the considered systems, this may be infeasible. For example, a networks made 10 switches having each 8 ports may lead to 80 single faults, more than 6.000 double faults, etc. Then, the approach considered here is to compute on demand, online a new configuration when some faults occur.
- *embedded computation*: The computation must be done with the resources embedded in the vehicle, no access to outside computation resources (cloud computing) is assumed.
- *short reconfiguration time*: after a fault, of course, the sooner the recovery, the better it is. Note that this reconfiguration can be decomposed into three sub-phases: *fault detection*, the identification of faulty elements, *computation*, the computation of a new configuration,

and the *deployment* of the computed configuration. Among these three phases, the deployment phase as specific issues, presented in Section 6.

- *continuous operation*: The computation and the deployment of the new configuration must penalise as little as possible the part of the system that is still in nominal mode, i.e. the data flows that do experience any fault on their path.

Several kinds of fault can appear on a system. In this study, we address permanent fail-silent faults of links, input port or output port. That is to say, a physical output port can be decomposed into a logical input port, and each sub-part can fail silently independently. Since the failure of a link is equivalent as the failure of the connected ports, only port failure will be considered.

Routing errors, transient faults, and babbling idiots will not be considered.

The fault detection is out of the scope of the challenge (it is assumed that the CNC is able to have this knowledge), even if the real solution will have to face it[1].

3 Case study

The case study consists of a TSN network with 5 switches (SW1 to SW5) connecting 15 avionics computers (ES1 to ES15). The computers belong to different avionics systems, either critical or non-critical.

Critical avionics systems are essential for the safe operation and navigation of an aircraft. Failure of these systems can have serious implications for flight safety. We consider following critical avionics systems and computers in the case study:

- Flight Control Systems
  - Autopilot System: Automates the control of the aircraft, maintaining heading, altitude, and speed. (ES1)
  - Fly-by-Wire System: Electronic interfaces that replace traditional manual flight controls. (ES2)
- Navigation Systems
  - Global Positioning System (GPS): Provides precise location and time information. (ES3)
  - Instrument Landing System (ILS): Assists with landing by providing lateral and vertical guidance. (ES4)
- Monitoring and Management Systems
  - Flight Management System (FMS): Integrates navigation, performance, and aircraft operation to provide operational efficiency. (ES5)
  - Engine Indication and Crew Alerting System (EICAS): Displays engine and system status and alerts the crew to issues. (ES6)
  - Aircraft Condition Monitoring System (ACMS): Monitors various parameters of the aircraft's systems and components for maintenance purposes. (ES7)
- Display System
  - Primary Flight Display (PFD): Shows essential flight data such as attitude, altitude, airspeed, and heading. (ES8)
  - Multi-Function Display (MFD): Can show a variety of information, including navigation maps, weather data, and system status. (ES9)

Non-critical avionics systems, while important for enhancing the functionality and convenience of the aircraft, are not essential for the basic operation and safety of the flight. We consider following non-critical avionics systems in the case study:

- Weather Information System

[1] The Connectivity Fault Management protocol (CFM, [8]) has been developed with a similar objective, but it does not fully comply with TSN. For example, Continuity Check(CC) messages are sent periodically to check the continuity between a source and a destination. But this just tests if there exists *one* path between the source and the destination. But in TSN, two flows from the same source to the same destination may use different paths (for load balancing for example), and the CC protocol will not be able to detect if one is broken.

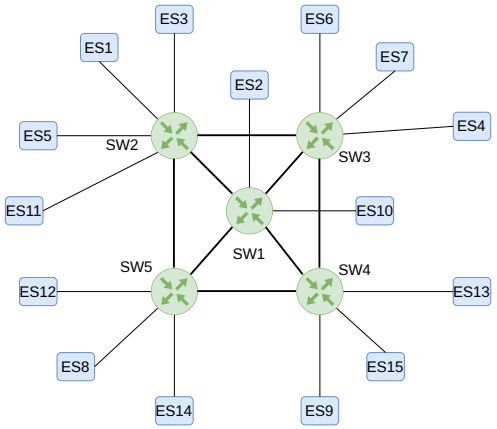

Figure 2: Case study: TSN network connecting avionics computers

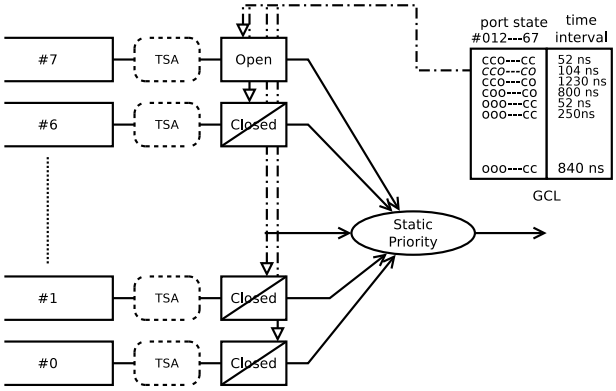

Figure 3: Architecture of a TSN output port.

- Provides real-time weather updates and forecasts to the flight crew, though not as critical as primary weather radar systems. (ES10)

- Cabin Management Systems
  - Control lighting, temperature, and other comfort-related features in the passenger cabin. (ES11)
  - Manage public address (PA) systems and intercoms for crew communication. (ES12)

- In-Flight Entertainment System
  - Provides passengers with audio and video entertainment options, games, and information about the flight. (ES13)

- Automatic Flight Information Reporting System
  - Transmits aircraft performance and maintenance data to ground stations for analysis and tracking. (ES14)

- Wi-Fi and Connectivity System:
  - Provide internet access and communication services for passengers and crew. (ES15)

4 TSN reminder

The Time-Sensitive Networking (TSN) group of IEEE has defined a set of addenda to Ethernet standard designed to provide a real-time network. The term TSN is widely used to name a network implementing these extensions. This paragraph presents a short overview of TSN, a very clear and complete presentation can be found in [4].

A TSN output port is made of (up to) 8 queues, called *traffic classes*, numbered from #7 to #0. When several queues compete for the output port, a static priority arbitration is used, #7 having the highest priority and #0 the lowest (cf. Figure 3). We do not consider preemption in this study.

To each traffic class is associated a "Transmission selection algorithm" (TSA), which is sometimes called "shaper". Only the "Credit-based shaper" (CBS), will be presented in this study[2]. Each traffic class is also controlled by a gate, which is open or closed, depending on the clock value and a configuration table (the Gate Control List – GCL). The GCL can be seen as a cyclic time schedule. This schedule states when each of the 8 gates is open or closed at each instant. It is implemented as a list of open/close states associated with a duration.

At any given time, the frame at head of a queue can compete for emission if its TSA allows it and if

[2]A presentation of ATS and CQF can be found respectively in [5] and [11].

its gate is open, and if it has enough time to be emitted up to completion (including any media dependent overhead like Inter Frame Gap, IFG [1, Fig. 8–17]).

4.1 TAS

The gates and the associated GCL have been introduced in [3] to forward *scheduled traffic* (ST), i.e. traffic where the transmission time of each frame respects a predefined schedule (also known as Time-Triggered – TT – scheduling [9]). But the implementation relies on gate schedule, not on frame schedule, adding some specific constraints when designing a schedule. Designing a GCL to ensure TT scheduling for a class is commonly called Time Aware Shaping (TAS), and the class is also called a TAS class.

- When the gate of one queue is open, the others may also be open, leading to some competition, resolved by the static priority arbitration and some optional preemption mechanism [2]. This is often avoided by building GCL such that, when the gate of a TAS class is open, all other gates are closed. This is called *exclusive gating*.
- Frames are stored in a FIFO queue. When a frame is received before another one, it must be forwarded before. No overtaking (order inversion) between frames is allowed (whereas it was possible in a frame-based schedule like TTEthernet [13]). The schedule has to deal with it.
- Since some jitter may occur in internal switching time, and since clocks are not perfectly synchronized, some margin must be accounted for in the schedule.
- When the gate is open, the head of queue frame will be forwarded (if the opening window is large enough).

Building a TAS schedule consists in configuring the GCL of each port such that each gate opening interval (commonly called "time window", "transmission window") is aligned with the one of the upstream and downstream port, in order to offer a bounded latency. An illustration is provided in Figure 5. An overview can be found in [14].

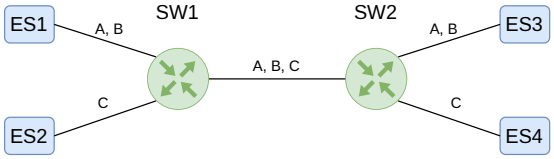

Figure 4: Simple network used to illustrate TAS behavior.

Also note that the last rule (forwarding the head of queue) differs from others Time-Triggered solutions (like TTEthernet [13]) not using queuing but whiteboard semantics (then selecting a frame based on its flow identification, not its place in a queue). As illustrated in Figure 5, this may lead to forward a frame even if this frame was not designed to use this window (e.g. if the frame that was supposed to use this window has been lost). This fragility has been identified in [6], and one proposed protection measure is the *frame isolation* constraint. It imposes that at any time, there is at most on frame in the TAS queue (which is not the case in the schedule of Figure5).

4.2 Credit Based Shaper (CBS)

A CBS queue is handled with a *credit* counter and an *idleSlope* parameter. The head of queue frame can be forwarded only if the credit value is non negative. The evolution rules of the credit are the following.

- When a frame is transmitted, the value of the credit is decreased by the size of the frame[3].
- When the credit is negative, it increases with slope *idleSlope*.
- When a frame is waiting whereas the gate is open (because of non preemption, or because some higher priority frame is using the output port, or because there is no enough time to send the frame before the next gate closing event), the credit increases with slope *idleSlope*.

[3]The behaviour described in the standard is slightly different. The credit first decreases with some *sendSlope* during frame emission, then increases with the idleSlope, but both behaviours are equivalents.

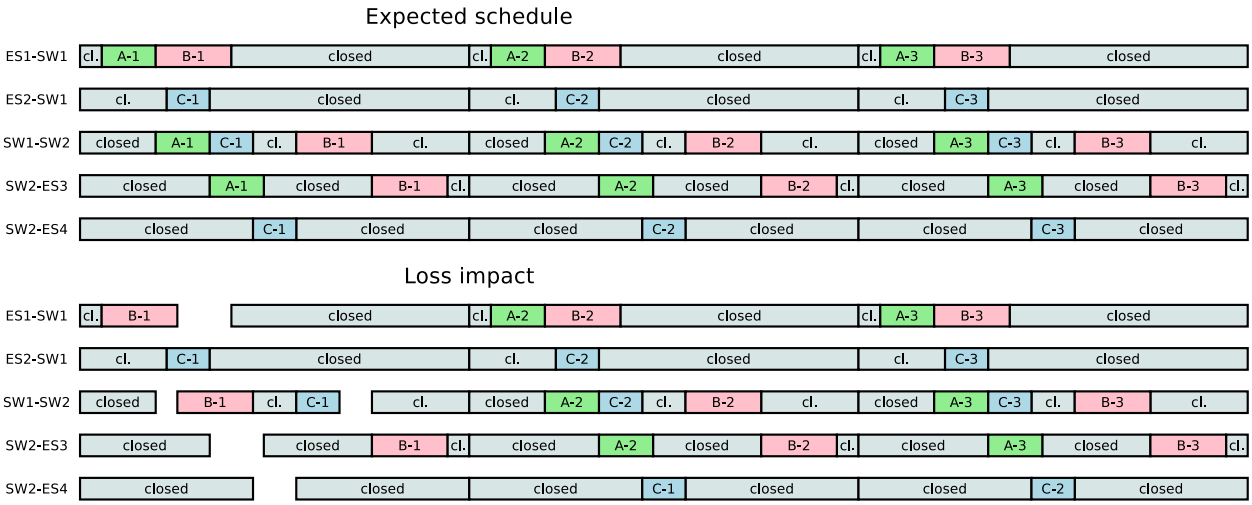

Figure 5: Illustration of GCL-based schedule for the network represented in Figure 5. This schedule assumes null switching time. Two flows, A and B, go from ES1 to ES3, and flow C goes from ES2 to ES4. Flows A and B share the same time window on link ES1-SW1, and flows A and C share the same time window on link ES1-SW2. If the frame A-1 is not sent by ES1, B-1 is forwarded and uses the time window reserved to A-1 and C-1, forcing C-1 to use the window of C-2, and so on.

- When the gate is close, the credit is frozen.
- When the queue is empty and the credit is positive, the credit is set to 0.

This behavior is illustrated in Figure 6. A discussion on evolution rules and TAS/CBS integration can be found in [7].

5 System general hypotheses

Since the TSN standard allows for a large set of configurations, here is the subset of mechanisms that are used in the initial configuration of the use case. More details will be provided in Section 3.

The system is a TSN network. Each switch has 8 queues/classes per output port. All ports do not have the same bandwidth.

On each switch port, the highest priority class, #7, is devoted to Time Aware Shaper (TAS). We assume *exclusive gating* and frame isolation.

TAS offers very small latencies and jitter, but its configuration is quite complex, especially when the number of flows increases [14]. Moreover, CBS can offer "good enough" latency guarantees for large set of data flows. Then, queues #6, #5 and #4 will be shaped using CBS. The other queues have no shaping mechanism and are used to host flows without time constraint.

This configuration will not use Cyclic Queuing and Forwarding (CQF) nor Asynchronous Traffic Shaping (ATS).

No preemption will be considered.

The set of flows is statically defined and fixed (in nominal mode). The routing is fixed and set at design.

The real system will use the Frame Replication and Elimination for Reliability (FRER) to increase robustness of the network, but in the context of this challenge, FRER will not be considered for application flows since it does not raise any specific problem.

In [1, § 46], TSN defines three kinds of configu-

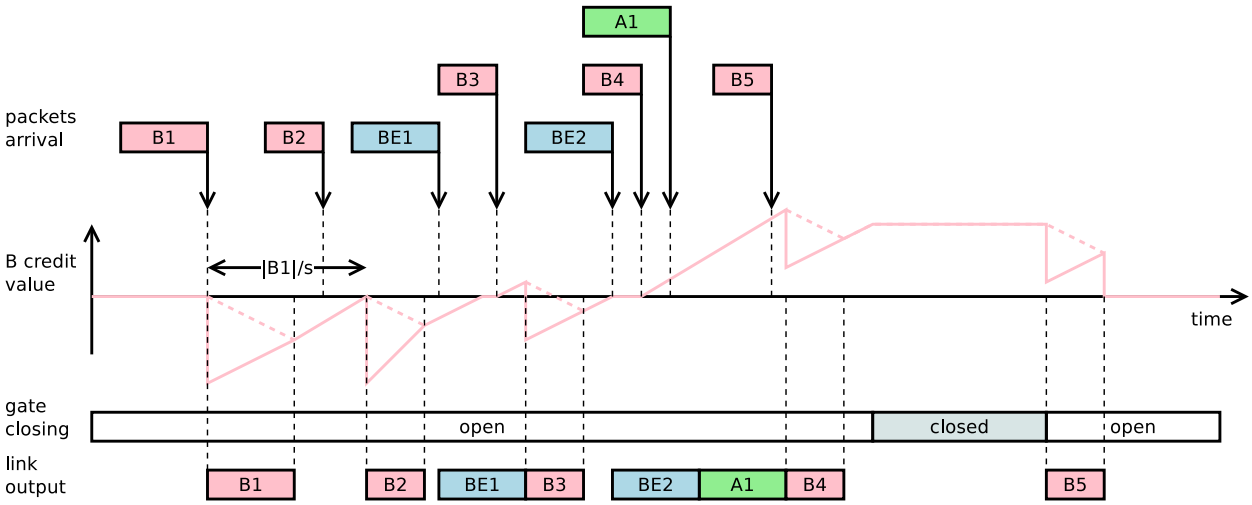

Figure 6: Example of the evolution of the CBS credit of a class B. The credit value following the rules presented is drawn in plain style, whereas the credit value as described by the standard is in dotted style. When frame B1 is received, the credit is null, and the frame can be forwarded. The credit is decreased by the frame size, —B1—, and then increases with slope s. When B2 is received, the credit is still negative, and B2 must wait. When a best-effort frame BE1 is received, the output link is idle, it can be forwarded. It then delay frame B3, and the credit becomes positive during this waiting time. Frame B4 is delayed by a best-effort frame BE2 and a higher priority frame A1. B5 can not be fully sent before the gate closing, so it has to wait. During gate closing, the credit is frozen.

ration architecture: fully decentralised, centralised network/distributed user and fully centralized. In the centralised network/distributed user architecture, each end-station is in charge of communicating its requirements to the network (and such requirements are centralised in a Centralised Network Configuration element – CNC), while in the fully centralised, a Centralised User Configuration element (CUC) is in charge of gathering all requirements, using some method, and this CUC exchange information with the CNC.

This system will consider the fully centralized architecture. Both the CNC and the CUC are deployed on (the same) switch, so the communication between the CNC and the CUC is considered fault-free. Conversely, FRER is used for the communications between the end-stations and the CUC, and between the CNC and the bridges.

Future evolution may consider CNC/CUC redundancy.

6 Configuration deployment issues

One issue in reconfiguration is the consistency problem. It can be easily illustrated with routing. Consider Figure 7: if a flow f_1 was using a link from SW0 to SW1, and if it must use a different route in the new configuration (either because some link on its initial route is down, or in order to decrease the load on a link it was using), using link (SW0,SW1), then at some time, the server SW0 must stop forwarding to SW1 and then forward to SW2. If SW2 is not aware of this change of route, and has no output port, it may drop the frame. One solution, presented in [12] is to replace f_1 by f'_1, and stop emitting f_1 once f'_1 has been added in all routing tables. But dropping frame is not the worst problem in TSN update. Consider the same example in the TAS case: in the new GCL schedule of SW1, there is no time slot devoted to f_1. Then, if the schedule is updated when there is a frame of flow f_1 in the queue, this frame will use the window of another flow, whose frame will be delayed to the next window, and so on, breaking the schedule. It may to the same kind of situation as in Figure 5.

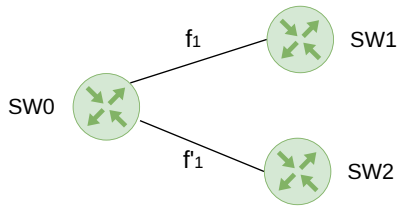

Figure 7: Routing update: flow f_1 being replaced by flow $f'1$.

This problem may be a rare event, but any solution to avoid it is welcome.

7 Computing a new configuration

The core of the challenge consists in computing a new configuration that provides the maximum *utility*. This notion of utility is part of the solution. Admitting the maximum number of flows is not an adequate criterion, since a TSN flow can host flows with high criticality for the system (control/command) and also less critical ones (entertainment). One may consider that priority (between #7 and #0 in TSN) is a utility function and that all flows with the same priority have the same utility. But the real-time community knows that priority allocation is one parameter to enhance the schedulability of a system [10]. Then, a solution considering a utility function independent of the priority will be valuable.

Computing a new configuration means selecting the flows, computing the routes, the classes, and the parameters.

Selecting the flows has two aspects. First, for some flows, it is "less worst" to have a degraded QoS than no access to the network (such degradation is specified by the system designer). Consider a flow f with some deadline D. If the algorithm is not able to build a configuration that satify this deadline, it may be better to propose a configuration that satisfy some $D' > D$ than rejecting f. Second, it is admissible to

drop some flow f' to allocate the associated resources to the flow f if is has a higher utility. Of course, a simpler solution where no degradation is considered, and no lower-utility flows are dropped may be proposed.

The route of each flow must be computed.

Each flow must be allocated to one TSN class. A solution is to keep the flow class (a flow in TAS class remains in TAS class, or is dropped, and the same for CBS), but a solution that change the flow class may be also of interest. For TAS class, the GCL schedule must be provided, and for CBS classes, the idleSlope must be provided.

The initial case study does not use CQF neither ATS, but any solution may use it if it comes with a strong benefit.

Likewise, the initial case study does not use Per-Stream Filtering and Policing (PSFP), but any solution may use it.

The objectives have been presented in Section 2, but they are not strong constraints. First, the objective to maintain continuous operation and to have short reconfiguration time are somehow contradictory: if some links are broken, the remaining links will be used to exchange data related to the new configuration, and these data may interfere with other ones. Second, the deployment of the new configuration may lead to some losses. As presented in Section 6, preventing a frame to disturb the scheduling may be a difficult task, and a solution may drop a few frames to start on a clean situation. In particular, it is better to drop one frame that to have a permanent shift of one cycle, as for flow C in the example in Figure 5.

References

[1] IEEE standard for local and metropolitan area networks – bridges and bridged networks. IEEE Standard 802.1Q, IEEE, 2022.

[2] IEEE standard for local and metropolitan area networks – bridges and bridged networks – amendment 26: Frame preemption. IEEE Standard 802.1Qbu, IEEE, 2016. https://doi.org/10.1109/IEEESTD.2016.7553415 doi:10.1109/IEEESTD.2016.7553415.

[3] IEEE standard for local and metropolitan area networks–bridges and bridged networks–amendment 25: Enhancements for scheduled traffic. IEEE Standard 802.1Qbv, IEEE, 2015. https://doi.org/10.1109/IEEESTD.2016.8613095 doi:10.1109/IEEESTD.2016.8613095.

[4] Lucia Lo Bello and Wilfried Steiner. A perspective on ieee time-sensitive networking for industrial communication and automation systems. *Proceedings of the IEEE*, 107(6):1094–1120, 2019.

[5] Marc Boyer. Equivalence between the urgency based shaper and asynchronous traffic shaping in time sensitive networking. *Leibniz Transactions on Embedded Systems*, 9(1):1–27, 2024.

[6] Silviu S. Craciunas, Ramon Serna Oliver, Martin Chmelík, and Wilfried Steiner. Scheduling real-time communication in IEEE 802.1Qbv time sensitive networks. In *Proceedings of the 24th International Conference on Real-Time Networks and Systems (RTNS'16)*, RTNS'16, pages 183–192, New York, NY, USA, 2016. Association for Computing Machinery. https://doi.org/10.1145/2997465.2997470 doi:10.1145/2997465.2997470.

[7] Hugo Daigmorte and Marc Boyer. Impact on credit freeze before gate closing in cbs and gcl integration into tsn. In *Proc. of the 27th Int. Conf. on Real-Time Networks and Systems (RTNS 2019)*, Toulouse, France, November 2019.

[8] IEEE. IEEE standard for local and metropolitan area networks – virtual bridged local area networks, amendment 5: Connectivity fault management. Technical Report 802.1Qag, IEEE, 2007.

[9] H. Kopetz. Event-triggered versus time-triggered real-time systems. In Arthur Karshmer and Jürgen Nehmer, editors, *Operating Systems of the 90s and Beyond*, pages 86–101, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg.

[10] Chung Laung Liu and James W Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)*, 20(1):46–61, 1973.

[11] Lisa Maile, Dominik Voitlein, Alexej Grigorjew, Kai-Steffen Hielscher, and Reinhard German. On the validity of credit-based shaper delay guarantees in decentralized reservation protocols. In *Proc. of the 31th Int. Conference on Real-Time Networks and Systems*, pages 108–118. Association for Computing Machinery, 2023. https://doi.org/10.1145/3575757.3593644 doi:10.1145/3575757.3593644.

[12] Zaiyu Pang, Xiao Huang, Zonghui Li, Sukun Zhang, Yanfen Xu, Hai Wan, and Xibin Zhao. Flow scheduling for conflict-free network updates in time-sensitive software-defined networks. *IEEE Transactions on Industrial Informatics*, 17(3):1668–1678, 2021. https://doi.org/10.1109/TII.2020.2998224 doi:10.1109/TII.2020.2998224.

[13] W. Steiner, G. Bauer, B. Hall, M. Paulitsch, and S. Varadarajan. TTEthernet dataflow concept. In *Proc. of Eighth IEEE International Symposium on Network Computing and Applications (NCA 2009)*, pages 319 –322, 2009. https://doi.org/10.1109/NCA.2009.28 doi:10.1109/NCA.2009.28.

[14] Thomas Stüber, Lukas Osswald, Steffen Lindner, and Michael Menth. A survey of scheduling algorithms for the time-aware shaper in time-sensitive networking (tsn). *IEEE Access*, 11:61192–61233, 2023. https://doi.org/10.1109/ACCESS.2023.3286370 doi:10.1109/ACCESS.2023.3286370.