CISTER
Research Centre in
Real-Time & Embedded
Computing Systems

Conference Paper

Implementation Cost Comparison of TSN Traffic Control Mechanisms

Aleksander Pruski
Mubarak Ojewale*
Patrick Meumeu Yomsi*
Michael Stübert Berger
Luís Almeida*

*CISTER Research Centre

CISTER-TR-210901

2021/09/10

Implementation Cost Comparison of TSN Traffic Control Mechanisms

Aleksander Pruski, Mubarak Ojewale*, Patrick Meumeu Yomsi*, Michael Stübert Berger, Luís Almeida*

*CISTER Research Centre

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8321159

E-mail: mkaoe@isep.ipp.pt, pmy@isep.ipp.pt, lda@fe.up.pt

https://www.cister-labs.pt

Abstract

The IEEE Time-Sensitive Networking (TSN) Task Group specifies a set of standards that enables real-time communication with predictable and bounded delays over the Ethernet. Specifically, TSN introduces a new set of so-called shapers, which regulate traffic arrival and transmission in the networks. Prominent among those are the IEEE 802.1 Qbv Time Aware Shaper (TAS) and IEEE 802.1Qav Credit-Based Shaper (CBS).Another traffic control mechanism is the IEEE 802.1Qbu Frame Preemption. Most works in the literature have focused on the quantitative performance comparison between these mechanisms. However, the discussion on how they compare in terms of implementation cost has received less attention. In this paper,we provide a comprehensive comparison of the implementation cost of the aforementioned TSN traffic control mechanisms. This comparison can help system designers in choosing which of the mechanism(s) to deploy for their applications

Implementation Cost Comparison of TSN Traffic Control Mechanisms

Aleksander Pruski
Comcores ApS, and
Technical University of Denmark (DTU),
Kgs. Lyngby, Denmark
Email: apr@comcores.com

Mubarak Adetunji Ojewale
CISTER Research Centre, ISEP
Polytechnic Institute of Porto, Portugal
Email: mkaoe@isep.ipp.pt

Voica Gavriluţ
Comcores ApS,and
Technical University of Denmark (DTU),
Kgs. Lyngby, Denmark
Email: vga@comcores.com

Patrick Meumeu Yomsi
CISTER Research Centre, ISEP
Polytechnic Institute of Porto, Portugal
Email: pmy@isep.ipp.pt

Michael Stübert Berger
Technical University of Denmark (DTU),
Kgs. Lyngby, Denmark
Email: msbe@fotonik.dtu.dk

Luis Almeida
CISTER Research Centre and
DEEC / FEUP University of Porto, Portugal
Email: lda@fe.up.pt

***Abstract*— The IEEE Time-Sensitive Networking (TSN) Task Group specifies a set of standards that enables real-time communication with predictable and bounded delays over the Ethernet. Specifically, TSN introduces a new set of so-called *shapers*, which regulate traffic arrival and transmission in the networks. Prominent among those are the IEEE 802.1 Qbv Time Aware Shaper (TAS) and IEEE 802.1Qav Credit-Based Shaper (CBS). Another traffic control mechanism is the IEEE 802.1Qbu Frame Preemption. Most works in the literature have focused on the quantitative performance comparison between these mechanisms. However, the discussion on how they compare in terms of implementation cost has received less attention. In this paper, we provide a comprehensive comparison of the implementation cost of the aforementioned TSN traffic control mechanisms. This comparison can help system designers in choosing which of the mechanism(s) to deploy for their applications.**

I. Introduction

A recent survey of communication solutions in industry [1] shows that Ethernet is leading the race in several real-time embedded and cyber-physical systems domains. The IEEE 802.1 Time-Sensitive Networking (TSN) Task Group, the successor of the Audio-Video Bridging (AVB) Task Group, developed a set of amendments and standards to add real-time features to Ethernet. With these features, it is possible to transmit data (signals, actuation, packets/frames) over a TSN-enabled Ethernet network with predictable and bounded latency & jitter. An important subset of these features is the set of so-called *shapers*, which enforce bandwidth reservations to improve predictability in the behavior of Ethernet frames.

These shapers are of two types: (1) *time-triggered* shapers; and (2) *event-triggered* shapers. The time-triggered shapers require that network nodes are synchronized and transmission decisions are based on pre-computed time schedules. Such shapers are suitable for applications with tight timing constraints. The Time Aware Shaper (TAS) (defined in IEEE 802.1Qbv [2]) is the most prominent time-triggered shaper.

978-1-7281-2989-1/21/$31.00 ©2021 IEEE

The time-synchronization requirement of time-triggered shapers is not needed in some real-time systems. Indeed, there are many real-time systems with unsynchronized devices. The event-triggered approach is often adopted in this case. Usually, event-triggered shapers provide only bounded delay. When used, the transmission of a frame is expedited if the network conditions warrant such. This is in contrast with the transmission of the time-triggered approach. The conditions, which could necessitate an expedited transmission of frames, will be discussed in detail in the later sections. Among event-triggered shapers described in AVB [3] and TSN, Credit Based Shaper (CBS) (defined by the IEEE 802.1Qav amendment [4]) is the most common. Another notable example for TSN is the Asynchronous Traffic Shaper (ATS) [5].

On another front, the IEEE 802.1Qbu amendment [6] defines the Frame Preemption feature, usually limited to only two traffic classes as specified in that amendment. A detailed description of the frame preemption mechanism is presented in Section II-B3. This mechanism improves the predictability of traffic by reducing the interference that urgent frames experience from long, lower priority frames.

System designers are interested in achieving acceptable performance on one hand, and in lowering the cost of hardware by choosing simpler and less complicated solutions on the other. Most of the studies in the literature that have compared the aforementioned traffic control approaches have done so mostly from the performance standpoint [7]–[10], focusing on the worst-case delay and jitter guarantees. However, the detailed discussion on how these mechanisms compare in terms of implementation cost is still missing. This discussion is important as it would help system designers in choosing which of the feature(s) to deploy for their applications. This paper aims at filling this gap by providing the implementation cost comparison of TSN traffic control mechanisms.

As long as performance requirements are met, businesses will opt to use the cheapest solution. The precise methods of cost estimation are beyond the scope of this work, however the

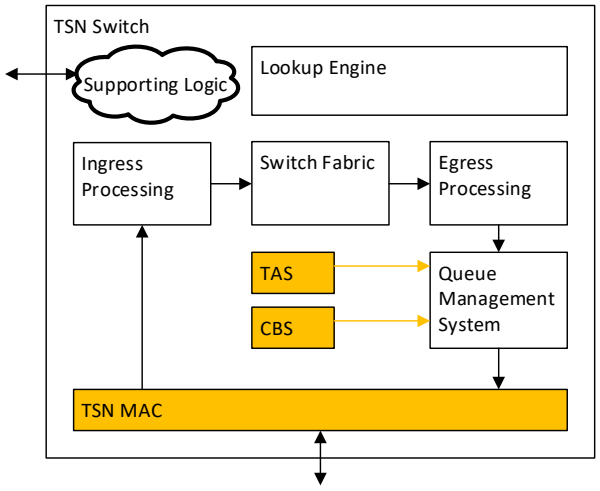

Fig. 1. Example block diagram of a TSN switching system

base classification is explained here. In a nutshell, equipment vendors use the term *non-recurring engineering (NRE)* for one-time costs, and *recurring engineering (RE)* for costs that repeat for every device (e.g. testing). In this context, the most suitable metric to measure implementation complexity is the sum of NRE costs, so surveying vendors for the cost of development of different TSN products would be ideal. However, due to the sensitive nature of this information, vendors are not willing to disclose it. Instead, we will focus on metrics that are easier to obtain, such as Field Programmable Gate Arrays (FPGA) resource utilization, described further in Section III. This metric is based on the assumption that more resources used means more costly implementation, thus approximating the NRE costs.

The rest of the paper is organized as follows. Section II provides context by describing the relevant TSN traffic control features. Section III provides an introduction to hardware implementations, metrics and methodology used. Section IV presents the results and Section V outlines related work. Finally, Section VI concludes this paper.

II. TSN Background

In this section we provide a concise introduction to TSN networks, followed by a short description of the TSN features considered for evaluation. We close it with a presentation of network configuration models used in TSN.

A. TSN network

A TSN network consists of end systems and TSN switches. Both are generally referred to as nodes. An end system can be any device that sends and/or receives data over the network, such as an Engine Control Unit, Camera, Radar, or Lidar. Switches are intermediate nodes between end systems, and they usually do not generate traffic. Any two nodes are connected by a full-duplex link, that is characterized by *speed* – the maximum rate at which it can transmit data.

1) TSN Switch: A TSN enabled switch is an Ethernet switch that supports one or more TSN features. In this work, we study in-depth three forwarding features: (1) the Credit-Based Shaper (CBS), (2) the Time-Aware Shaper (TAS) and (3) Frame Preemption. In the block diagram presented in Fig. 1, the frames flow through the TSN MAC (supporting frame preemption), are processed in the ingress processing block, then moved to the egress processing block via the switch fabric. After that, they are queued at the output and await servicing. This is where the TSN shapers provide control data to the queue management system. When selected for transmission, they exit through the egress processing path of the TSN MAC.

In addition to being a well-designed piece of hardware, a TSN switch has to be configured appropriately to meet the traffic requirements. This configuration process falls under the responsibility of Control and Management (C&M) planes. The exact description of C&M planes can be found in [11]. According to this document, the control plane "instructs network devices (...) how to process and forward packets"; while the management plane "monitors, configures and maintains" ports of the network node. The data plane contains all functions responsible for handling the traffic directly.

In Fig. 1, the TAS and CBS blocks exchange the control information with the queue management block, influencing when the frames are dequeued. This can be directly mapped to the transmission selection function of the forwarding process from IEEE 802.1Q. Correct operational parameters must be set apriori to ensure consistent shaping across the network. Since the shaping functions do not depend on the transmitted data itself, they can be classified as belonging to C&M planes.

2) TSN traffic: Data traffic in networks can be usually divided into flows, where a flow[1] is a sequence of frames sent from a source end-system to one or more destination end-systems, through a set of TSN switches. Generally, flows can be further classified as either *periodic* or *aperiodic* [12]. In TSN networks, the emphasis is placed on periodic flows, for which *flow-instances* (a single frame or a small burst) are produced recurrently at the source, and are separated by a predefined time interval (*period*).

Flows in TSN networks must also meet timing constraints. The most common one of those is the *deadline*, which imposes a maximum *end-to-end delay* for all frames belonging to a flow. Another common requirement is the *end-to-end delay variation*, commonly referred to as *jitter*.

A multitude of flows can exist in a network, but treating each of them differently would require a fine classification resolution, and therefore high (and expensive) processing capabilities in each network node. Consequently, in TSN, the flows with similar Quality of Service (QoS) and timeliness requirements are grouped in traffic classes. In IEEE 802.1Q, there exist up to eight traffic classes. Each of the aforementioned flows must be mapped to a traffic class. At each switch,

[1]Terms *flow* and *stream* are used interchangeably in the context of TSN.

flows are further mapped to Internal Priority Values (IPV), and then to actual egress queues.

B. TSN Features

Having discussed the basics of TSN networks and their requirements in the previous section, we can proceed with a more detailed description of TSN features relevant to this work, namely: CBS, TAS, and Frame Preemption.

1) CBS: To provide bounded end-to-end delays, the Audio-Video Bridging (AVB) Task Group (TG) of the IEEE introduces the CBS – specified in IEEE 802.1Qav [4]. The purpose of CBS is to guarantee an appropriate share of the link bandwidth to each traffic class, as well as to *shape* – spread traffic bursts over time. In contrast with Strict Priority, CBS prevents the flows of the high-priority classes from starving out those of lower priority classes. It defines the *credit* parameter to regulate the traffic forwarding at the egress queues.

The selection for transmission of a head-of-queue frame is conditioned by the following: (1) the value of credit is at least zero and (2) there are no other higher priority frames available for transmission. The fluctuation of credit is influenced by the (i) *idle slope* configuration parameter and (ii) *sending slope* parameter (derived from (i)). The credit evolves as follows:

1) The credit is initially zero.
2) It decreases with sending slope during transmission.
3) The credit increases with idle slope when frames are waiting, but higher priority frames are transmitted.
4) If the queue becomes empty while the credit is positive, it is reset to zero.

The idle slope, as defined in IEEE 802.1Qav, represents a fraction of link speed, and the sending slope is computed as the difference between idle slope and link speed.

2) TAS: To ensure low delay and jitter for time-sensitive traffic, the TAS is specified in IEEE 802.1Qbv. TAS imposes independent time windows by opening and closing the gates that are associated with each queue of an egress port. The opening and closing times are stored in the so-called Gate Control List (GCL). When the channel is free, the next frame is selected for transmission if: (1) belongs to the queue with the highest priority and (2) the queue's gate is open.

To ensure a fully deterministic transmission of time-sensitive traffic with TAS, for each flow, the time intervals have to be aligned along the path. This alignment can be obtained only when all nodes have the same notion of time, i.e., when their clocks are synchronized. The clock synchronization is a complex mechanism that is defined for TSN-based networks in IEEE 802.1AS [13] standard. This standard leverages a profile of IEEE precision time protocol (PTP) 1588-200 [14] called the generic PTP (gPTP) to synchronize the clocks in a distributed network through a master-slave architecture. The IEEE 802.1AS first uses the Best Master Clock (BMC) algorithm to select the node in the network with the most accurate clock. Such a node is called GrandMaster (GM), and, after selecting the GM, all network nodes are continuously adjusting their clocks to the GM's one. For more details about BMC selection and clock adjustments please see the IEEE 802.1AS standard.

In addition to the fact that TAS requires time synchronization, the GCLs have to be provided, their synthesis being also a very computationally intensive task. To obtain a fully deterministic transmission with TAS, it is required that the opening of gates along the path of each critical flow be carefully determined. Currently, the main approach is to use offline pre-computed GCLs. This is a very rigid approach though, which is unsuitable for dynamic changes of the network and of the traffic. Another approach is to update the pre-computed GCLs. For both approaches, a centralized network configuration model (see Section II-C) is required. Such a model is needed since the global view of the network is required for the GCLs synthesis.

3) Frame Preemption: Frame preemption was defined and introduced in the IEEE 802.3br [15] and IEEE 802.1Qbu standards [6]. These specify a one-level preemption scheme for Ethernet frames. In this scheme, two Medium Access Control (MAC) sub-layer service interfaces are defined in the link layer: a preemptable MAC (pMAC) interface and an express MAC (eMAC) interface. Frames assigned to the eMAC and pMAC service interfaces are referred to as "*express*" and "*preemptable*" frames respectively. Every eMAC frame is assigned a higher priority than any pMAC frame and can thus preempt any of these at almost any time instance. Frames of the same class cannot preempt each other. The behavior of Frame Preemption is as follows. At every TSN switch output port where this feature is enabled, a higher priority frame can suspend the transmission of a lower priority frame that has already started, prior to its completion, in order to transmit through the same port. In this case, the transmission of the preempted frame would resume only after all the higher priority frame(s) that have arrived during the frame suspension window are fully transmitted. Those operations are usually executed in a separate MAC Merge Sublayer (MMS), which then provides eMAC and pMAC with non-fragmented frames. While the mode of operation of frame preemption is different from the shapers, the objectives of both approaches are the same: to achieve deterministic transmission delays. In terms of configuration, the assignment of queues to either the eMAC or pMAC needs to be decided. By default, the higher priority queues are assigned to the eMAC.

While frame preemption improves the performance of express frames, the network performance is negatively impacted in scenarios where the number of express frames is high. Another limitation is the fact that preemptable frames with timing requirements can suffer long blocking periods due to the non-preemptive service of frames in the same category. This is irrespective of the individual priority level of each frame. Recently, a multi-level preemption scheme has been proposed to circumvent these limitations [16], [17]. Under such a scheme, more preemption classes can exist (in contrast to just two classes defined in the standards), and time-sensitive preemptable frames can preempt other lower-priority preemptable frames.

C. Network-wide Control and Management

The performance that can be achieved with the TSN features discussed above depends strongly on how well the network is configured. In this sub-section, we briefly discuss the network configuration and management mechanism used in TSN.

1) Initial Approach: Traditionally, in AVB, the network control was distributed. This means that the activities of control plane, as required for CBS, were done locally by each switch. The Stream Reservation Protocol (SRP), specified in IEEE 802.1Qat [18], deals with traffic advertisement and resource reservation on an end-to-end basis.

Briefly, the reservation procedure is as follows. When a switch receives for the first time the *stream advertisement message*, it checks the available resources of each port. If there are ports that have available resources, the stream advertisement message is marked accordingly and forwarded. A port has available resources for a stream if (a) there is available bandwidth and (b) it is estimated that, by traversing this switch, the stream still meets its deadline. Bandwidth availability is rather intuitive, while the deadline check is more complex – involving the estimation of the delay to reach and traverse this switch. This estimation uses the already allocated bandwidth and the idle slope. After a while, the switch receives the advertisement message back. This time it contains additional information returned by the end-stations interested in receiving the stream. Then the switch decides whether to accept and forward the stream.

2) TSN Novel Approach: With the improvements proposed through TSN features, some configurations should be synchronized across the network and the distributed configuration model is no longer enough. For example, with the original SRP, a switch was only allowed to decide whether to forward a stream or not, but there were no mechanisms to align the idle slopes across the network. Therefore, the IEEE 802.1Qcc [19] amendment proposes enhancements for the SRP. One of the most important contributions of this amendment is the introduction of three network configuration models, namely, fully distributed, centralized network/ distributed user, and fully centralized models. For the fully distributed model, the reservation requests are carried out by User Network Information (UNI) messages. According to the specification [19], the TSN features, such as frame preemption and TAS, can be configured only with the centralized network configuration models.

The centralized network/ distributed user model introduces the concept of Centralized Network Configuration (CNC). Here, end systems send UNI to the edge-switches — the switches that are directly connected to the end systems. The edge-switches forward these UNIs to the CNC. Furthermore, CNC, using remote management, discovers network topology and switches capacities. Based on this general view of the network, the CNC decides the resource allocation for each flow. Finally, the CNC, using again the remote management, conveys the updated resource allocation to the switches.

Finally, the fully centralized configuration model introduces the concept of Central User Configuration (CUC). There are application domains, such as automotive or industrial automation, where the data production or consumption has to be synchronized with data transmission. For such cases, the CUC is used for discovering the end systems and their traffic demands and capacities. Based on this, CUC generates the UNIs and sends them to the CNC. CNC determines now the configuration for switches and end systems and conveys the updated configuration to switches. Moreover, the CNC sends back to the CUC the configuration for end systems. Furthermore, it is the job of CUC to convey the configurations to the end systems.

III. Hardware development and comparison metrics

We recall, that the main goal of this paper is to compare the implementation complexity and, by extension, the development cost of switches implementing different TSN features. For this purpose we employ resource utilization reports of hardware modules executing TSN functions. Those modules are implemented in Xilinx Field Programmable Gate Array (FPGA) devices. The reports are obtained during digital design process outlined in Section III-A, and contain counts of basic hardware blocks (described in Section III-B) necessary for implementation. Section III-C then describes how those numbers are used for comparison with their inherent limitations.

A. Digital design process

The TSN modules are implemented as Register Transfer Level (RTL) Silicon Intellectual Property (IP) Cores. The source code of those IP cores, written in a Hardware Description Language (HDL), is then processed by Computer Aided Design (CAD) tools [20] in order to:

- express the design with logic gates (*logic synthesis*),
- map obtained logic circuit to components specific for a given implementation technology and then place and route them on either a silicon wafer or a subtype of Programmable Logic Device (PLDs) (*physical design*).

One of the outputs from the *physical design* is the *size* of the implemented module. In Application Specific Integrated Circuit (ASIC) workflow, the physical design maps the logic circuit to cell libraries, and the end result for size is just a surface area on a wafer. In FPGAs, which are a sub-type of PLD, the devices already have a fixed number of resources that the circuit is mapped to. The size is then expressed as a number and type of resources used.

B. Xilinx FPGAs

At a high level, a basic FPGA device consists of Logic Blocks, Programmable Interconnect and I/O pins [20]. In Xilinx devices, the main resource used to implement both sequential and combinatorial circuits, are Configurable Logic Blocks (CLBs). Those blocks contain Look-Up Tables (LUTs) (an n-input truth table) for combinatorial logic and storage elements for sequential logic (CLB Registers). The CLBs also contain dedicated carry logic for arithmetic operations (CARRY8) and Multiplexers (FnMUX) to maximize resource utilization within one block. For more information on FPGA

resources, as well as on modern FPGA architecture, please refer to [21], [22].

In addition to CLBs, some FPGA devices contain specialized blocks and hard IPs (non-programmable modules fulfilling a specific function, e.g. transceivers). Among those specialized blocks are:

- Block Random Access Memories (BRAM), containing arrays of Static RAM (SRAM) cells and sometimes FIFO logic,
- Digital Signal Processing (DSP) blocks, dedicated for executing more complex arithmetic operations,
- clock tiles, containing primitives for clock generation and buffering.

All of our synthesis runs were executed targeting Xilinx Zynq UltraScale+ family, device `xczu9cg-ffvb1156-1-e`, since this device is used in popular ZCU102 boards.

C. Comparison methodology and context

To accurately contextualize the cost added by each TSN feature, we have chosen an absolute baseline to be the whole `xczu9cg-ffvb1156-1-e` device. A switching system similar to the one in Fig. 1, however without any TSN features, could also be used for that purpose. Since its size can vary significantly for different architectures and configurations, the whole FPGA device is a more firm reference point.

We also compare TSN modules with each other. This comparison is enabled by a weakly-coupled and modular design. In such a design, the queuing and buffering systems (located in *queue management system* in Fig. 1) are common for TAS and CBS. Therefore the memories and logic in them can be excluded from the comparison. Contrasting Frame-preemption and TSN shapers poses a different challenge. The TSN MAC must possess some buffering capabilities in the RX direction, so it can reassemble frame-fragments. The size of those buffers depends on maximum frame size, however we assume it to be minimal when compared to the buffers in the *queue management system*. The buffers in the TSN MAC are therefore also excluded from the comparison.

Obtained results have the drawback of only approximating the NRE costs by capturing the end-result *size* of the product. They do not capture the RE costs associated with testing every single chip. This approximation is enabled by the assumption, that the bigger the device, the longer and harder it is to develop. This assumption contradicts the fact that the optimization phase of development is aimed at decreasing the area/resources. If significant development effort is placed in that phase, the result is a *smaller* device. Nevertheless, within one organization, the level of optimization is expected to be similar, enabling us to use the resource utilization reports for our comparison.

IV. Implementation and Results

In this section the implementations of CBS, TAS, and (Multi-level) Frame Preemption [16] are described, and associated resource utilization numbers are given. The implementations themselves are proprietary, therefore they are presented only at the overview level.

A. CBS & TAS Implementations

A block diagram further decomposing the TAS and CBS blocks from Fig. 1 is presented in Fig. 2. The *configuration register bank* (separate for TAS and CBS) contains all required run-time configurable registers and interface implementation, allowing both TAS and CBS modules to be managed by external software. The CBS controller implements the CBS algorithm and performs credit bookkeeping based on the current configuration and transmission status provided by the queuing system. CBS provides the *transmission allowed* flag for each supported priority, which is then used by the queuing system to select frames for transmission. In order to manage credit, CBS requires information about current *queue state* and *transmission status.*

The *TAS controller* implements the functions described in Section II. TAS provides the *gate state* and *gate guard* control signals to the queue management system. The *gate guard* control signal enables the usage of fixed guard-bands (as defined in Annex Q of IEEE 802.1Q [23]). This implementation of the guarding feature calculates how much time the gate will remain open, which is necessary to prevent transmission window overrun errors.

Additionally, TAS contains a memory holding Gate Control Lists (GCL). By default, this memory is mapped to BRAMS. However, we can precisely estimate its size in bits based on the number of supported priorities and number of supported entries. This estimation is shown in Equation 1, where M is the memory size in bits, n is the number of supported GCL entries, $\mathrm{TI}_{\mathrm{width}}$ is the bit-size of the time-interval entry, and N_{IPVs} is the number of supported priorities.

$$M = 2 \times n \times (\mathrm{TI}_{\mathrm{width}} + N_{\mathrm{IPVs}}) \tag{1}$$

Finally, TAS requires a synchronized Time of Day (ToD) to be available in the system. This is visualized by the *tod provider* entity in Fig. 2. The overhead (in terms of resource

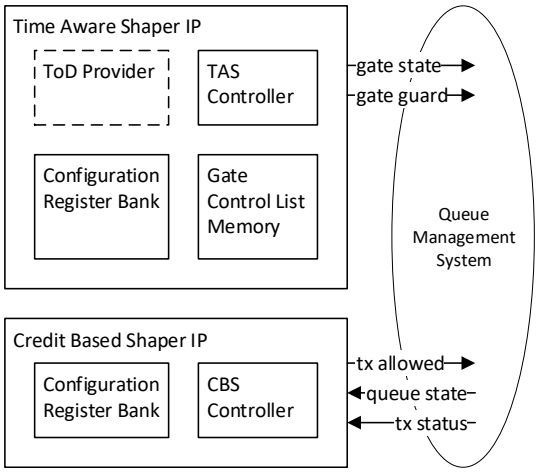

Fig. 2. TAS and CBS block diagram

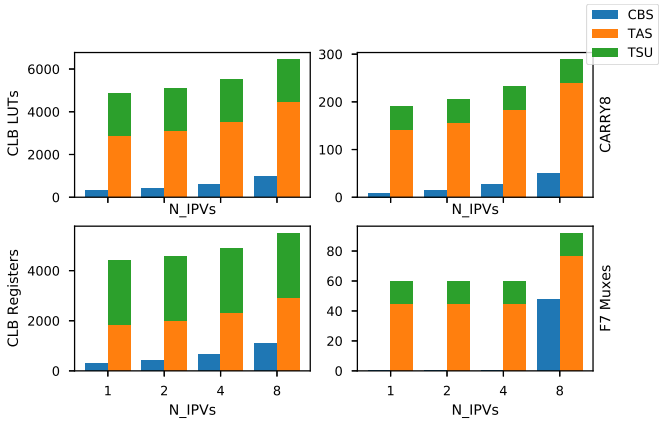

Fig. 3. CBS and TAS CLBs

utilization) of this subsystem is included in our results for TAS, and marked as Time Stamping Unit (TSU).

B. Resource utilization for TAS and CBS

For the TSN shapers, we performed synthesis runs for 1, 2, 4, and 8 supported priorities (N_IPVs). The corresponding counts for LUTs, Registers, CARRY8 and F7 Muxes are shown in Fig. 3. As one can observe, LUTs, Registers, and CARRY8 adders scale linearly with the number of supported priorities. This is explained by the fact that both credit keeping modules in CBS and guarding modules in TAS are instantiated per priority. When deployed together TAS and CBS influence each others' operation, however we observed no significant additional overhead in such scenario.

Table I provides the percentage of total resources available on `xczu9cg-ffvb1156-1-e` used by TAS IP (in rows marked *%TOT*), and also the overhead of TAS over CBS (how much more resources it uses, rows marked */CBS*). As expected, the TAS is significantly larger, with overhead being as high as 15 times larger in case of only one supported priority. A significant portion of this can be attributed to the TSU (as seen in Fig. 5), together with the previously explained guarding feature (not visualized). What is worth noticing, is that the LUT utilization of TAS is around 2%. With an instance of TAS required per egress port, it makes scaling it with port numbers especially expensive. Since F7 MUXes only start being used in CBS with 8 priorities, and their utilization numbers are low, they are omitted from Table I.

C. Frame Preemption implementation

A simplified block diagram of multi-level frame preemption system is shown in Fig. 4. A single-level frame preemption is fully encompassed by *TSN MAC* module. As mentioned in Section II, it consists of eMAC, pMAC, and MMS. The multi-level preemption could be naively implemented with additional MAC and MMS instances [16]. This is shown in Fig. 4 where the *supporting logic* encompasses everything necessary for the integration of the modules, however not directly implementing any functionality. Predictably, the naive

TABLE I
TAS IP RESOURCE UTILIZATION AND OVERHEAD OVER CBS

N_IPVs	Type	CLB LUTs	CLB Registers	CARRY8
1	%TOT	1.8%	0.8%	0.6%
	/CBS	1512%	1500%	2122%
2	%TOT	1.9%	0.8%	0.6%
	/CBS	1187%	1116%	1366%
4	%TOT	2%	0.9%	0.7%
	/CBS	913%	763%	863%
8	%TOT	2.4%	1%	0.8%
	/CBS	663%	500%	567%

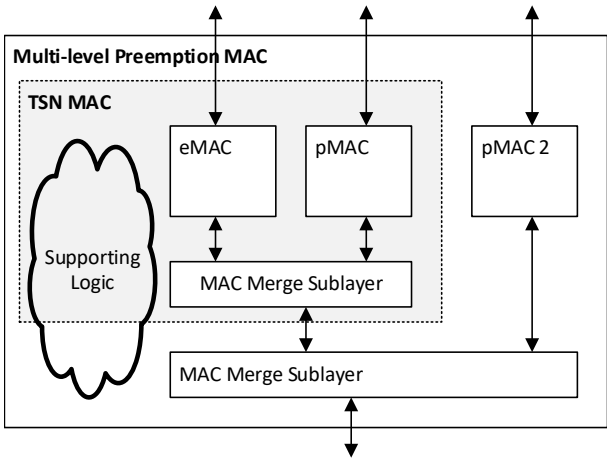

Fig. 4. (Multi-level) Frame Preemption block diagram

approach causes resource doubling for two-level preemption, tripling for three-level, and so on.

In Fig. 5 stacked bar plots are used to visualize the resource utilization of main sub-modules of the TSN MAC. In Table II, the percentage of total resources available on the target device is listed in *%TOT* rows. Since the increase in utilization is linear, only preemption for zero, one, and two levels is visualized. Additionally, in rows marked *%OH*, the overhead of implementation over the non-preemption mac (row 0) is listed. The values are exclusive of supporting logic. The MMS implements more complex operations of preemption, which explains its size being more than twice of the plain MAC.

For two traffic classes, one-level frame preemption would be preferred over TAS, if it can meet the performance requirements. If more traffic classes must be supported however, the choice is not so clear-cut. The described naive implementation approach can still be used, and three-level preemption will provide four distinct classes. With the observed scaling factor, the size will be comparable with TAS, however going to eight classes would make multi-level preemption significantly more expensive than TAS. It does not account however for additional software required to achieve time synchronization.

V. Related work

There is a plethora of research that compares the TSN traffic control approaches from a performance standpoint. On this

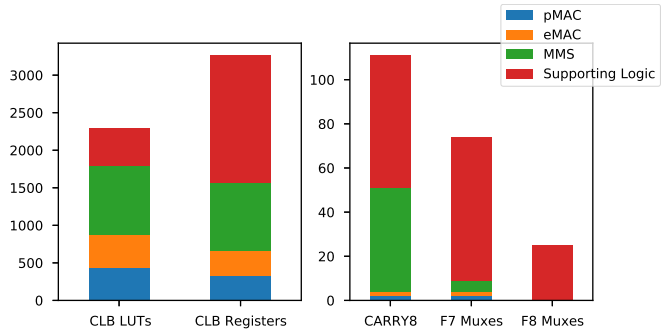

Fig. 5. Resource utilization for TSN MAC

TABLE II
FRAME PREEMPTION RESOURCE UTILIZATION AND OVERHEAD

Pre. levels	Type	CLB LUTs	CLB Registers	CARRY8	F7 Muxes
0	%TOT	0.16%	0.06%	0.01%	0.00%
1	%TOT	0.66%	0.28%	0.15%	0.01%
	%OH	411.21%	467.37%	2550.00%	450.00%
2	%TOT	1.15%	0.51%	0.29%	0.01%
	%OH	723.11%	835.03%	5000.00%	800.00%

front, Thangamuthu et al. [24] compare the performance of three TSN shapers, namely, TAS, bandwidth limiting shaper and the Peristaltic Shaper (PS). In this work, the comparison metrics are delay and jitter. The authors conclude that TAS provides the best performance for both metrics. A similar work by Thiele et al. [25] compares TAS with PS, reaching a similar conclusion, i.e., TAS provides better end-to-end delay performance. They noted, however, that the performance of TAS degrades significantly when end systems are not synchronized. Thiele et al. [8] also compare the performance of TAS and frame preemption and conclude that the two approaches achieve very similar end-to-end delays. The authors go further to suggest, that standard Ethernet with frame preemption is a viable alternative to TAS.

Another work comparing TAS and frame preemption, by

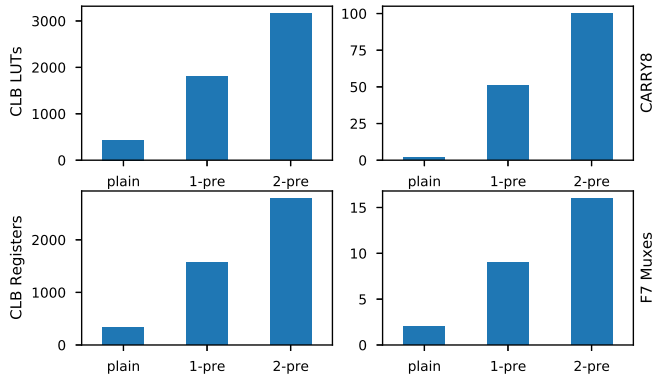

Fig. 6. Resource increase caused by adding preemption levels.

Gogolev and Bauer [7], demonstrated via experiments that frame preemption is more suitable for industrial networks with unsynchronized end systems. The authors point out that a serious limitation for frame preemption is the low Quality of Service (QoS) resolution , i.e., there are only two classes (eMAC and pMAC) specified for preemption in TSN. To mitigate the aforementioned limitation, Ojewale et al. [16], [26] introduce the multi-level preemption paradigm. Here, the authors also compare TAS and frame preemption from a qualitative standpoint and promote the multi-level frame preemption feature as a viable alternative to TAS. Furthermore, Knezic et al. [27] provided an enhanced implementation approach for the multi-level preemption feature and the formal worst-case traversal time analysis of this approach was provided by Ojewale et al. [26]. On another front, a work by Nasrallah et al. [10] conducts a performance comparison of TAS and Asynchronous Traffic Shaping (ATS). The authors surmise that ATS performs well compared to TAS for sporadic (asynchronous) traffic. They noted, however, that the performance of Scheduled Traffic degrades with increasing Best-Efforts load under ATS. We note that none of the aforementioned works have compared these features from an implementation complexity perspective.

There is not such intense interest though in comparing the performance of CBS with other shapers. There are the works of Alderisi et al. [28] and Lo Bello [29] that present simulation results for AVB-ST (Scheduled Traffic) in industrial automation and automotive areas, respectively. AVB-ST is a stern approach of TAS, assuming that the scheduled traffic is strictly periodic and that precise offsets are provided for the scheduled frames. In these works, the baseline is considered the pure CBS, and the authors observe that only by using AVB-ST the bounded delay and zero jitter for the scheduled traffic is ensured. A similar work, by Meyer et al. [30], highlights the importance of quality of schedule tables. For example, for a non-dense table, the delay of AVB traffic increases but is still bounded and compliant with the specification in [3]. On the other hand, when using a compact schedule table, the delays of AVB traffic are still bounded, but not compliant anymore with the specification.

On the hardware implementation front, Zhou et al. [31] present a VHDL design layout for the transmission and reception processes as well as an FPGA-based hardware implementation of TSN sender and receiver nodes. Hotta et al. [32] present another FPGA implementation of an Ethernet switch supporting frame preemption.

VI. CONCLUSION

In this work, we compare the implementation cost of three TSN features: TAS, CBS and frame preemption. The aim of this comparison is to guide system designers on the choice of the TSN feature to adopt, provided that the temporal performance requirements are met. The cost comparison is based on FPGA resource utilization for each of the features.

The related works show that TAS gives the best performance in terms of timeliness. But, from our results, we also observe

that TAS has the highest starting resource utilization among the three features, thereby making it the most expensive to implement. It is even more so, if costs of achieving time synchronization are included. The CBS is the cheapest to implement. For frame preemption, while the cost is significantly higher than that of CBS, it is very much lower than that of TAS. If multi-level preemption is implemented, the cost grows linearly. It is less than TAS for up to four levels of preemption, after which it overtakes TAS. We note that this projection does not include the additional overhead of time synchronization software, which is required by TAS.

For future work, comparison of the implementation cost of more TSN flow control features can be considered. Here, the features such as ATS and Cyclic queuing and forwarding are the prime candidates. Additionally, the overhead of integrating these features together is also worth investigating. Finally, extending the comparison for different FPGA vendors and device families would make the results more generalizable.

ACKNOWLEDGEMENTS

This work is partially supported by Comcores ApS and Innovationsfonden Denmark through grants 8053-00031B and 9066-00048B; and also by National Funds through FCT/MCTES (Portuguese Foundation for Science and Technology), within the CISTER Research Unit (UIDB/04234/2020).

REFERENCES

[1] B. Akesson, M. Nasri, G. Nelissen, S. Altmeyer, and R. I. Davis, "An empirical survey-based study into industry practice in real-time systems," in *IEEE Real-Time Systems Symposium (RTSS)*, 2020.

[2] IEEE, "IEEE standard for local and metropolitan area networks – bridges and bridged networks - amendment 25: Enhancements for scheduled traffic," *IEEE Std 802.1Qbv-2015 (Amendment to IEEE Std 802.1Q—as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd-2015, and IEEE Std 802.1Q—/Cor 1-2015)*, pp. 1–57, 2016.

[3] IEEE, "IEEE standard for local and metropolitan area networks– audio video bridging (avb) systems– corrigendum 1: Technical and editorial corrections," *IEEE Std 802.1BA-2011/Cor 1-2016 (Corrigendum to IEEE Std 802.1BA-2011)*, pp. 1–13, 2016.

[4] AVB Task Group, "IEEE 802.1Qav - Forwarding and Queuing Enhancements for Time-Sensitive Streams," 2009.

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

[6] IEEE, "IEEE standard for local and metropolitan area networks – bridges and bridged networks – amendment 26: Frame preemption," *IEEE Std 802.1Qbu-2016 (Amendment to IEEE Std 802.1Q-2014)*, pp. 1–52, 2016.

[7] A. Gogolev and P. Bauer, "A simpler tsn? traffic scheduling vs. preemption." in *25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, 2020, pp. 183–189.

[8] D. Thiele and R. Ernst, "Formal worst-case performance analysis of time-sensitive ethernet with frame preemption," in *IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2016, pp. 1–9.

[9] D. Hellmanns, J. Falk, A. Glavackij, R. Hummen, S. Kehrer, and F. Dürr, "On the performance of stream-based, class-based time-aware shaping and frame preemption in tsn," in *IEEE International Conference on Industrial Technology (ICIT)*, 2020, pp. 298–303.

[10] 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.

[11] E. Haleplidis, K. Pentikousis, S. Denazis, J. H. Salim, D. Meyer, and O. Koufopavlou, "Software-Defined Networking (SDN): Layers and Architecture Terminology," RFC 7426, Jan. 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc7426.txt

[12] G. C. Buttazzo, *Hard Real-Time Computing Systems : Predictable Scheduling Algorithms and Applications*. Springer, 2011.

[13] IEEE, "IEEE standard for local and metropolitan area networks - timing and synchronization for time-sensitive applications in bridged local area networks," *IEEE Std 802.1AS-2011*, pp. 1–292, 2011.

[14] ——, "IEEE standard for a precision clock synchronization protocol for networked measurement and control systems," *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, pp. 1–300, 2008.

[15] ——, "Standard for ethernet amendment 5: Specification and management parameters for interspersing express traffic," *Std 802.3br-2016 (Amendment to IEEE Std 802.3-2015)*, pp. 1–58, Oct 2016.

[16] M. A. Ojewale, P. M. Yomsi, and B. Nikolić, "Multi-Level Preemption in TSN: Feasibility and Requirements Analysis," in *2020 IEEE 23rd International Symposium on Real-Time Distributed Computing (ISORC)*, May 2020, pp. 47–55.

[17] M. A. Ojewale, P. Meumeu Yomsi, and G. Nelissen, "On multi-level preemption in ethernet," in *WiP Session (ECRTS)*, 2018, pp. 16–18.

[18] AVB Task Group, "IEEE 802.1Qat - Stream Reservation Protocol," 2010.

[19] TSN Task Group, "IEEE 802.1Qcc - Stream Reservation Protocol (SRP) Enhancements and Performance Improvements," 2018.

[20] S. D. Brown and Z. G. Vranesic, *Fundamentals of Digital Logic with Verilog Design*, 3rd ed. New York: McGraw-Hill Higher Education, 2014.

[21] Xilinx, "UltraScale Architecture Configurable Logic Block User Guide (UG574)," 2017.

[22] ——, "UltraScale Architecture and Product Data Sheet: Overview (DS890)," 2020.

[23] IEEE 802.1 Working Group, "IEEE 802.1Q-2018 - IEEE Standard for Local and Metropolitan Area Networks," 2018.

[24] S. Thangamuthu, N. Concer, P. J. Cuijpers, and J. J. Lukkien, "Analysis of ethernet-switch traffic shapers for in-vehicle networking applications," in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2015, pp. 55–60.

[25] D. Thiele, R. Ernst, and J. Diemer, "Formal worst-case timing analysis of ethernet TSN's time-aware and peristaltic shapers," in *Proceedings of the IEEE Vehicular Networking Conference (VNC)*, 2015, pp. 251–258.

[26] M. A. Ojewale, P. M. Yomsi, and B. Nikolić, "Worst-case traversal time analysis of tsn with multi-level preemption," *Journal of Systems Architecture*, vol. 116, p. 102079, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1383762121000667

[27] M. Knezic, M. Kovacevic, and Z. Ivanovic, "Implementation Aspects of Multi-Level Frame Preemption in TSN," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, Sep. 2020, pp. 1127–1130.

[28] Alderisi, Giuliana and Patti, Gaetano and Lo Bello, Lucia, "Introducing support for scheduled traffic over IEEE audio video bridging networks," in *IEEE Conference on Emerging Technologies Factory Automation (ETFA)*, 2013, pp. 1–9.

[29] Lo Bello, Lucia, "Novel trends in automotive networks: A perspective on Ethernet and the IEEE Audio Video Bridging," in *IEEE Emerging Technology and Factory Automation (ETFA)*, 2014, pp. 1–8.

[30] Meyer, Philipp and Steinbach, Till and Korf, Franz and Schmidt, Thomas C., "Extending IEEE 802.1 AVB with time-triggered scheduling: A simulation study of the coexistence of synchronous and asynchronous traffic," in *IEEE Vehicular Networking Conference (VNC)*, 2013, pp. 47–54.

[31] Z. Zhou, Y. Yan, S. Ruepp, and M. Berger, "Analysis and implementation of packet preemption for time sensitive networks," in *IEEE 18th International Conference on High Performance Switching and Routing (HPSR),*, 2017, pp. 1–6.

[32] Y. Hotta, A. Inoue, H. Bessho, C. Mangin, and R. Kawate, "Experimental study of a low-delay ethernet switch for real time networks," in *16th IEEE Int. Conf. on High Performance Switching and Routing*, 2015.